Programming technic of JavaScript that can be used at once.

JavaScript テクニックブック

古籏 一浩 / 著

JavaScript technic book

```
window.onload = function(){

document.getElementById("ch
eckButton").onclick =
function(){

document.getElementById("re
sult").innerHTML =
browser.type()+"<br>";

document.getElementById("re
sult").innerHTML +=
browser.OS();
 }
```

```
var browser = {
 type : function(){
  if (window.opera) return "Opera";
  if (window.createPopup) return "Internet
Explorer";
  if (navigator.userAgent.indexOf("Safari") > -1)
return "Safari";
  if (navigator.userAgent.indexOf("Firefox") > -
1) return "Firefox";
  return "unknown";
 },

window.onload = function(){
 var root = document.getElementById("myfLink");
 var aTag = root.getElementsByTagName("a");
 for (var i=0; i<aTag.length; i++){
  if (aTag[i].className.match(/ctrlWindow/)){
   aTag[i].onclick = function(){
    window.opener.location.href = this.href;
    return false;
   }
```

```
window.onload = fu
 var hMenu1 = new
Spry.Widget.Menu   ("MenuBar1");
}
window.onload = fu  ction(){
 setInterval("album  swapImage()",
2000);
}
var album = {
 imageURL : [
    "images/01.jpg",
```

最新トレンドのワザが満載！

Ajaxの台頭によって俄然注目を浴びているJavaScript。ライブラリを使った見せ場たっぷりのスゴ技を実践的な例題でわかりやすく解説。

24時間無料でサンプルデータをダウンロードできます

C&R研究所

Programming technic of JavaScript that can be used at once.

JavaScript テクニックブック

古籏 一浩 / 著

JavaScript technic book

```
window.onload = function(){

document.getElementById("ch
eckButton").onclick =
function(){

document.getElementById("re
sult").innerHTML =
browser.type()+"<br>";

document.getElementById("re
sult").innerHTML +=
browser.OS();
 }
```

```
var browser = {
 type : function(){
  if (window.opera) return "Opera";
  if (window.createPopup) return "Internet
Explorer";
  if (navigator.userAgent.indexOf("Safari") > -1)
return "Safari";
  if (navigator.userAgent.indexOf("Firefox") > -
1) return "Firefox";
  return "unknown";
 },

window.onload = function(){
 var root = document.getElementById("myfLink");
 var aTag = root.getElementsByTagName("a");
 for (var i=0; i<aTag.length; i++){
  if (aTag[i].className.match(/ctrlWindow/)){
   aTag[i].onclick = function(){
    window.opener.location.href = this.href;
    return false;
   }
```

```
window.onload = function() {
 var hMenu1 = new
Spry.Widget.MenuBar("MenuBar1");
}
window.onload = function(){
 setInterval("album.swapImage()",
2000);
}
var album = {
 imageURL : [
      "images/01.jpg",
```

C&R研究所

■権利について

■本書の内容について

- 本書は著者・編集者が実際に操作した結果を慎重に検討し、著述・編集しています。ただし、本書の記述内容に関わる運用結果にまつわるあらゆる損害・障害につきましては、責任を負いませんのであらかじめご了承ください。
- 本書で紹介している各操作の画面は、Windows Vista（日本語版）とInternet Explorer7を基本にしています。他のOSやブラウザをお使いの環境では、画面のデザインや操作が異なる場合がございますので、あらかじめご了承ください。
- 本書に掲載しているサンプルは、Internet Explorer5.5以降、Firefox1.5以降、Safari2以降、Opera9以降で動作します。旧バージョンのブラウザでは動作対象外となりますので、ご了承ください。
- Internet Explorer7の場合、ローカル環境上での動作とサーバー上にアップロードしたときの動作が異なっています。ローカル環境上では期待通りに動作しないため、サーバー上にアップロードして動作を確認してください。

■サンプルデータについて

- 本書のサンプルデータは、C&R研究所のホームページからダウンロードすることができます。ダウンロード方法については、17ページを参照してください。
- サンプルデータの動作などについては、著者・編集者が慎重に確認しております。ただし、サンプルデータの運用結果にまつわるあらゆる損害・障害につきましては、責任を負いませんのであらかじめご了承ください。
- サンプルデータの著作権は、著者及びC&R研究所が所有します。許可なく配布・販売することは堅く禁止します。

●本書の内容についてのお問い合わせについて

この度はC&R研究所の書籍をお買いあげいただきましてありがとうございます。本書の内容に関するお問い合わせは、FAXまたは郵送で「書名」「該当するページ番号」「返信先」を必ず明記の上、次の宛先までお送りください。お電話や電子メール、または本書の内容とは直接的に関係のない事柄に関するご質問にはお答えできませんので、あらかじめご了承ください。

〒950-3122 新潟県新潟市北区西名目所4083-6　株式会社 C&R研究所　編集部
FAX 025-258-2801
『JavaScriptテクニックブック』サポート係

はじめに

JavaScriptが登場して10年以上が経過しましたが、その間にJavaScriptも少しずつ進化を遂げてきました。近年、JavaScriptが再度注目されるようになったのは、非同期通信を利用して各種処理を行うAjax(Asynchronous JavaScript + XML)の普及によるものです。

Ajaxの普及によってJavaScriptが注目されるようになったのは好ましいことですが、他言語を利用しているプログラマから見たJavaScriptとWebページ/サイト制作を行ってきたWebプログラマから見たJavaScriptでは見方が異なり、扱いも異なっています。最近、JavaScriptを始めたプログラマはロジック(アルゴリズム)は正しいのに動作しない、または特定のブラウザでしか動作しない、という問題に悩まされるでしょう。ロジックとしてはきれいなプログラムであっても現実的に動かなければ、どうしようもありません。現実的にロジックとしては美しくなくても、多くのブラウザで確実に動作するプログラムの方がよいのです。

しかし、いつまでも旧来の手法のままでは時代に合わなくなり、行き詰まってしまうことがあります。たとえば、HTML文書とスタイルシート、JavaScriptコードは分離されているべきだといわれますが、書籍ではこれらを分離して解説したものはあまり見当たりません。また、急にコードを分離せよと求められても、どのように分離しコードを書けばよいのか困惑してしまいます。

そこで、本書では多く見かける処理に対してHTML、スタイルシート、JavaScriptコードを分離して掲載しています。これまでのJavaScriptの解説書(Webサイトなど)で書かれているコードと本書に書かれているコードを比較して見るとよいでしょう。新しい書き方に移行する際に役立つはずです。

2007年4月

古籏 一浩

本書の読み方

本書の各ページのレイアウトは、次のようになっています。

●見出し
この項目の内容を一言で表しています。

●サンプルデータの所在
ダウンロードしたサンプルデータのフォルダ階層を表しています。ダウンロード方法は17ページを参照してください。

●導入文
この項目の概要を説明しています。

●ファイル名
サンプルのファイル名を表しています。

●章タイトル
この章のタイトルを表しています。

●SOURCE CODE
サンプルのソースコードを掲載しています。

●ソースコードの解説
ソースコードの重要な箇所を解説しています。番号はソースコード内の番号に対応しています。

●折り返し記号
サンプルのソースコードが改行されていないことを表しています。

CHAPTER-03 ▶ 027

SECTION 027 ページが読み込まれたら指定したテキストフィールドにフォーカスを移す

ここでは、ページの読み込み時に、指定したテキストフィールドにフォーカスを移す方法について解説します。

JavaScript Sample - Windows Internet Explorer

IDとパスワードを入力してログインしてください。

ID:
PW:
ログイン

ページが表示されると、指定したテキストフィールドにフォーカスが移動する

CHAPTER-03 フォーム

SOURCE CODE　HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>IDとパスワードを入力してログインしてください。</h1>
    <form action="./check.cgi" method="get" name="mainForm">
      ID:<input type="text" name="userID" id="userID"><br> ― 1
      PW:<input type="text" name="userPW" id="userPW"><br>
      <input type="button" id="loginButton" value="ログイン">
    </form>
  </body>
</html>
```

1 …フォーカスしたいエレメントにID名を指定する

72

最新情報について

本書の記述内容において、内容の間違い･誤植･最新情報の発生などがあった場合は、「C&R研究所のホームページ」にて、その情報をいち早くお知らせします。

URL http://www.c-r.com （C&R研究所のホームページ）

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("userID").focus(); ―1
}
```

1 …「focus()」を使って指定したエレメントにフォーカスを移す

OnePoint フォーカスするには「focus()」を使う

ページ上の特定のエレメントにフォーカスを移すには、「focus()」を使います。ページが読み込まれたら特定のテキストフィールドにフォーカスするには、「window.onload」イベントが発生した際にフォーカス処理を行うようにします。サンプルではID名がuserIDのテキストフィールドにフォーカスするようにしています。

Column フォーカスを外す

フォームのテキストフィールドにフォーカスされた状態からフォーカスを外すには、「blur()」を使います。サンプルの場合は、次のように「focus()」の代わりに「blur()」を指定します。

```
document.getElementById("userID").blur();
```

Column テキストフィールドの内容を選択する

ページが読み込まれたらテキストフィールドの内容を選択するには、「focus()」と「select()」を組み合わせます。次のように最初にフォーカスし、その後に選択するようにします。

```
window.onload = function(){
  document.getElementById("userID").focus();
  document.getElementById("userID").select();
}
```


ページが表示されると、指定したテキストフィールドの内容が選択される

●インデックス
どの章かわかりやすいように見開きのページの両側にインデックスを配置しています。色が付いているところが該当の章で、上から順に1～12章を示しています。

●OnePoint
操作例のテクニックのポイントや仕組みを解説します。

●Column
発展的な情報を解説します。

CONTENTS

CHAPTER-01 JavaScriptの基礎知識

CHAPTER-02 基本テクニック

CHAPTER-03 フォーム

CHAPTER-04 画像

CHAPTER-05 ダイアログ/ウィンドウ

CHAPTER - 06 DOM

CHAPTER - 07 スタイルシート

CHAPTER - 08 イベント

CHAPTER - 09 文字列

CHAPTER-10 日付/時刻

CHAPTER-11 その他

CHAPTER-12 GUI関連

サンプルデータのダウンロード方法

本書のサンプルデータは、C&R研究所のホームページにある「データ館」というページからダウンロードすることができます。本書のサンプルを入手するには、次のように操作します。なお、本書のサンプルを入手する際は、18ページに記載してあるユーザー名とパスワードが必要になります。

■C&R研究所のホームページ■
URL http://www.c-r.com

1 C&R研究所のホームページを表示し、「データ館」をクリックします

2 本書の書籍名のリンクをクリックします

サンプルのダウンロードに必要なユーザー名とパスワード

ユーザー名 jstb

パスワード jat17

www.c-r.com に接続

Please enter username & password

ユーザー名(U): jstb

パスワード(P): *****

□パスワードを記憶する(R)

OK キャンセル

3 ユーザー名を入力します

4 パスワードを入力します

5 クリックします

TOPページ 会社案内 新刊案内 既刊本 龍之介の部屋 達人募集 データ館 読者の広場 譲ります 訂正情報

JavaScriptテクニックブック

<サンプルデータのダウンロードページ>

※目次を見るにはここをクリックしてください

CHAPTER-01 JavaScriptの基礎知識(js_01.zip/90KB)

CHAPTER-02 基本テクニック

CHAPTER-03 フォーム(js_03

CHAPTER-04 画像(js_04.zip

CHAPTER-05 ダイアログ/ウ

CHAPTER-06 DOM(js_06.zip/59KB)

開く(O)
新しいウィンドウで開く(N)
対象をファイルに保存(A)...
対象を印刷(P)
切り取り(T)
コピー(C)
ショートカットのコピー(T)
貼り付け(P)
お気に入りに追加(F)...
プロパティ(P)

6 目的の章のサンプルデータのリンクを右クリックし・・・

7 [対象をファイルに保存(A)]を選択します

名前を付けて保存

保存する場所(I): デスクトップ

マイ ドキュメント
マイ コンピュータ
マイ ネットワーク

最近使ったファイル
デスクトップ
マイ ドキュメント
マイ コンピュータ
マイ ネットワーク

ファイル名(N): js_01.zip

ファイルの種類(T): 圧縮 (zip 形式) フォルダ

保存(S) キャンセル

8 保存先を指定します

9 クリックします

サンプルデータはZIP形式で圧縮されています。パソコンのOSがWindows XPの場合は、ダウンロードしたサンプルファイル(「js_01.zip」など)を右クリックして[すべて展開(A)]を選択すると、解凍することができます。

CHAPTER - 01

JavaScriptの基礎知識

SECTION　KNOWLEDGE

001 JavaScriptの概要

JavaScriptとは

JavaScriptは手軽なスクリプト言語の1つです。JavaScriptは多くのブラウザに搭載されており、ページ上の画像やテキストの切り替え、非同期通信を使ったデータ処理などに利用されています。よく見かける画像のロールオーバーも多くはJavaScriptにより実現されています。

JavaScriptはInternet ExplorerやFirefox、Safariなどのブラウザに搭載されているため、ブラウザ専用のスクリプト言語と思われがちです。しかし、JavaScriptはWindowsやMacOS XなどのOS、Adobe Photoshopなどのアプリケーション、Adobe FlashやPDFなどにも搭載されており、幅広く利用されています。それだけ柔軟な言語仕様ともいえます。

JavaScriptの基本となる部分の仕様はECMAにより公開されており、ECMA Scriptとして呼ばれています。この基本部分にブラウザやアプリケーションが、状況に応じて独自のオブジェクトを追加したスクリプト言語が一般的にいわれるJavaScriptです。

また、マイクロソフト社の場合は、JavaScriptと互換性を持った独自拡張のスクリプト言語「JScript」としてブラウザなどに実装されています。JScriptではWindows固有のActive Xを利用した処理（ファイル処理などローカル環境の制御など）も可能になっています。

●ECMAのホームページ

▶プロトタイプ指向のオブジェクト指向言語

JavaScriptは、Javaなどの一般的なクラスを利用したプログラミング言語と異なり、基本となるオブジェクトを拡張/派生させるプロトタイプ指向のオブジェクト指向言語です。ただし、当初実装されたブラウザであるNetscape 2と現時点のものでは異なっている部分が多くあります。当初はプロトタイプによる継承もなく、単純なオブジェクト指向言語で変数名や文字数制限、利用できるオブジェクトなど多くの制限がありました。これが、次第に改良され現在に至っています。

▶Ajaxの普及とライブラリの登場

ブラウザの普及とともにJavaScriptも改良されましたが、ブラウザによって実装が異なり動作も異なってしまうことが多く、制作者泣かせの言語であることも確かです。これを解消するのがJavaScriptライブラリです（46ページ参照）。Ajaxの普及とともに急速にライブラリが整備されてきています。もっとも有名なのは、prototype.jsです。これはJavaScriptの言語特性を利用し各種メソッドを拡張し、よりオブジェクト指向のプログラムが作成できるライブラリです。しかし、MacOS X + Internet Explorer 5など少し古いブラウザでは読み込んだだけでエラーとなったり動作しないこともあります。ライブラリを利用すれば楽にはなりますが、少し前のブラウザに対応させるという要望が出た時点で破綻することもあります。

▶HTML/スタイルシートも含めてJavaScriptでプログラミングする

JavaScriptのプログラムが簡単になるか、複雑になるかはHTMLの構造に大きく依存します。このため、単純にHTMLコーダーが正しい文書構造を作成しても「正しいけど使えない、役立たず」になってしまうことがあります。これはHTMLだけでなく、スタイルシートにもいえます。スタイルシートで、どんなにうまくデザインしてもJavaScriptで制御する際には、仕様を変更したり修正しないと駄目なことがあります。

特にまずいのは、JavaScript側でスタイルの数値などを指定しなければならないといった状況です。このような状況が発生するとファイルはHTML、スタイルシート、JavaScriptと分かれているのに、デザインの修正はJavaScriptプログラマが行わなければならないという本末転倒な事態が派生してしまいます。この辺りは、制作者同士でよく検討すべきでしょう。

近年、Web標準仕様に沿ってページを制作するのが一般的になりましたが、HTML、スタイルシートに関してはさまざまな手法が開発され全体的なレベルも向上しました。しかし、JavaScript部分は意外と古いままの実装や考え方のまま使われているようです。そこで本書ではHTML、スタイルシート、JavaScriptを分離し、次のステップに進めるような踏み台のような感じでサンプルを用意し掲載しています。DOMスクリプティングを行っている人には、不満もあるかもしれませんが、次のステップに進むための参考になれば充分と考えています。

Column Ajaxとは

Google Mapsの登場で注目されるようになったのがAjax（「Asynchronous JavaScript + XML」を略した名称）です。これは、JavaScriptを使って非同期通信を行う処理を示します。これによりブラウザで何か処理を行うたびにページを切替えることがなくなり、より使いやすいユーザーインターフェースを提供できるようになりました。当初はAjaxの名前の通り、JavaScriptでXMLデータを処理するのが基本でしたが、現在はXMLデータだけでなく、JSON形式のデータが使われることもあります。これはXMLデータでは通信を行う際にデータ量が多くなり処理速度が低下してしまうため、軽量で手軽なJSON形式が使われるようになってきています。また、標準の非同期通信を行うオブジェクトはセキュリティの都合上、同一ドメイン上のサーバーしか通信ができません。しかし、JSON形式を利用することで、そのような制約を超えて他のサーバーからデータを受け取ることができます。これにより、さらに利用範囲が広がっています。

SECTION 002

KNOWLEDGE

JavaScriptの基本事項について

JavaScriptのプログラムコードの記述方法

JavaScriptプログラムのコードを記述するには、いくつかの方法があります。

▶HTML/XHTMLファイル内に記述する場合

HTML/XHTMLファイル内に記述する場合は、文書内にプログラムコードを書きます。基本的には、次のように<script>〜</script>タグ内に記述します。HTMLの場合は、<script>タグに対応していないブラウザでプログラムが表示されないようにHTMLのコメント「<!--」と「//-->」で囲みます。XHTMLの場合は、本来は別ファイルにするのがよいのですが、どうしても駄目な場合は「//<![CDATA[」と「//]]>」で囲みます。<script>タグは、特に<head>タグ内でなく<body>タグで囲まれた範囲でも問題なく記述できます。

```
<script type="text/javascript">
<!--
  ここにスクリプトを書く
//-->
</script>
```

```
<script type="text/javascript">
//<![CDATA[
  ここにスクリプトを書く
//]]>
</script>
```

古いブラウザにも対応させたい場合には、次のように「language」属性も指定します。

```
<script language="javascript" type="text/javascript">
<!--
  ここにスクリプトを書く
//-->
</script>
```

また、HTMLでデフォルトのスクリプト言語としてJavaScriptを指定する場合は、次の<meta>タグを指定しておきます。

```
<meta http-equiv="Content-Script-Type" content="text/javascript">
```

▶外部ファイルとして記述する場合

外部ファイルとして使用する場合は、純粋にスクリプトだけを記述したテキストファイルを作成し、<script>タグの「src」属性でファイル名を指定します。

```
<script src="ファイル名.js"></script>
```

ファイル名の末尾(拡張子)は「.js」にする必要があります。「src」属性には「.js」ファイルだけでなく、CGIのパスなどを記述し結果をJavaScriptコードで返すこともできます。また、JSON形式など、データだけを取得したい場合にも利用することができます。

▶タグ内に記述する場合

タグ内に記述する場合には、イベント名を記述し、「=」の後ろに実行するプログラムを指定します。多くの場合、関数を呼び出すようになっていますが、通常のプログラムコードをそのまま記述することができます。

```
<a href="#" onClick="alert('ok?')">~</a>
```

「onClick」や「onMouseover」などのイベント名は、すべて大文字で書いても小文字で書いても動作するため、大文字小文字が混在していても問題ありません。ただし、プログラムで定義する場合にはすべて小文字にする必要があるので、注意してください。

また、タグ内に直接プログラムコードを書くのは、好ましい状態ではありません。本書のサンプルのようにHTML文書とJavaScriptコードのファイルを分離するようにしてください。

行の記述

1行の区切りは、「;」(セミコロン)です。命令の途中で改行することも可能ですが、この場合ブラウザ側が自動的に行末かどうかを判別し処理します。スクリプト解釈エンジンとHTML解釈エンジンの都合によりスクリプト内に</script>の文字があると、以後に続くプログラムが表示されてしまうことがあります。このような場合には、「"<"+"/script>"」のように文字列として分割/結合して対処することができます。

また、命令にエラーなどは実行時にチェックされます。

JavaScriptで扱える数値

数値は10進数、8進数、16進数を扱うことができます。8進数は先頭が「0」で始まり、16進数は先頭が「0x」で始まります。小数を表す場合は「.」(ピリオド)を使います。負数は先頭に「−」(マイナス)を付加します。たとえば、10進数では「4649」「−2.718」、16進数では「0xfda5」「0x9AB4」、8進数では「011」などになります。

また、大きい数値や小さい数値の場合は、「e」または「E」を使って記述することもできます(指数の意味のE)。たとえば、「−3.1E12」「.1e12」「2E−12」のようになります。

文字列の扱い

文字/文字列は、「"」(ダブルクォーテーション)か「'」(シングルクォーテーション)で囲みます。たとえば、「"皆既日食"」「'2035年'」のように記述します。「"」で囲まれた文字列内に「"」を含める場合は「¥"」、「'」で囲まれた文字列に「'」を含める場合は「¥'」と指定します。たとえば、「"ダンクーカ¥""」「alert(¥'書いてやるぜ¥')」のように記述します。「¥」は使用する書体によっては「＼」(バックスラッシュ)になります。

ユニコード文字の指定(中国語やアラビア文字などを混在させる場合)は、「¥u文字列」のように「¥u」を指定します。

特殊文字

JavaScriptで用意されている特殊な文字には、次の文字があります(ここでは「¥」記号になっていますが、使用する機種とフォントによって「＼」(バックスラッシュ)になる)。これらの特殊文字の中には、期待通りに動作しない「¥b」「¥t」などがあります。「¥n」はアラートダイアログに表示する文字列を改行表示する場合に便利です(例:「alert("Sample¥nTest")」。

特殊文字	内　容
¥b	バックスペース
¥f	フォームフィード
¥n	改行
¥r	リターン
¥t	タブ
¥¥	¥記号
¥"	ダブルクオーテーション
¥'	シングルクオーテーション
¥NNN	8進数で指定
¥xNN	16進数で指定
¥uNNNN	ユニコードで指定

リテラル

リテラルとは、定数値です。「true」「false」や文字列である「"MZ-80K2E"」、数値の「123.4」などが該当します。「true」は数値の「0」以外、「false」は数値の「0」で代用することも可能ですが、なるべく「true」「false」を使用すべきです。

003 演算子について

JavaScriptで使える演算子

JavaScriptには基本的な演算子のみ用意されており、高度な演算を行う場合は「Math」オブジェクトを利用するようになっています。JavaScriptで使用できる演算子は、右表のようになります。シフトなどのビット演算は、32ビット長で処理されます。

演算子	意　味	記述例
+	加算	z = x + y
−	減算	z = x − y
＊	乗算	z = x ＊ y
/	除算	z = x / y
%	剰余	z = x % y
++	プリインクリメント	y = ++x（代入前に加算）
++	ポストインクリメント	y = x++（代入後に加算）
−−	プリデクリメント	y = −−x（代入前に減算）
−−	ポストデクリメント	y = x−−（代入後に減算）
-	符号反転	y = -x
&	論理積	15 & 9
&&	論理積（論理式）	(x == 12) && (y == 4)
|	論理和	15 | 9
||	論理和（論理式）	(x == 12) || (x == 9)
^	排他的論理和	15 ^ 9
!	否定	!(x == 12)
<<	符号付き左シフト	9<<2
>>	符号付き右シフト	9>>2
>>>	符号なし右シフト	19>>>2

代入演算子

JavaScriptには、代入演算子も用意されています。使用できる代入演算子は、下表のようになります。

代入演算子	記述例	「=」を使った記述
=	x = y	—
+=	x += y	x = x + y
−=	x −= y	x = x − y
＊=	x ＊= y	x = x ＊ y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
<<=	x <<= y	x = x << y
>>=	x >>= y	x = x >> y
>>>=	x >>>= y	x = x >>> y
&=	x &= y	x = x & y
^=	x ^= y	x = x ^ y
|=	x |= y	x = x | y

三項演算子

JavaScriptには、条件によりどちらか1つの値を取る三項演算子も用意されています。三項演算子は「?」と「:」(コロン)で区切って指定します。たとえば、変数「b」の値に応じて代入する文字列を変えるには、次のように記述します。

```
a = (b > 2) ? "2以上" : "2未満"
```

条件を満たした場合には「?」直後の値が代入され、条件を満たしていない場合には「:」より後ろの値が代入されます。

演算子の優先順位

演算子には、優先順位が決められています。優先順位は、下表のようになります。表の上にある演算子が優先順位が高くなり、下に行くに従って優先順位が低くなります。

<table>
<tr><th>優先順位</th><th>演算子</th><th>記　号</th></tr>
<tr><td>1</td><td>括弧</td><td>()　[]　.</td></tr>
<tr><td>2</td><td>符号反転、デクリメント、インクリメント</td><td>!　˜　−　++　−−</td></tr>
<tr><td>3</td><td>乗除算</td><td>*　/　%</td></tr>
<tr><td>4</td><td>加減算</td><td>+　−</td></tr>
<tr><td>5</td><td>ビットシフト</td><td><<　>>　>>></td></tr>
<tr><td>6</td><td>比較</td><td><　<=　>　>=</td></tr>
<tr><td>7</td><td>等価、不等価</td><td>===　==　!=</td></tr>
<tr><td>8</td><td>ビット演算/論理積</td><td>&</td></tr>
<tr><td>9</td><td>ビット演算/排他的論理和</td><td>^</td></tr>
<tr><td>10</td><td>ビット演算/論理和</td><td>|</td></tr>
<tr><td>11</td><td>論理積</td><td>&&</td></tr>
<tr><td>12</td><td>論理和</td><td>||</td></tr>
<tr><td>13</td><td>三項演算</td><td>?:</td></tr>
<tr><td>14</td><td>算術</td><td>=　+=　−=　*=　/=　%=　<<=
>>=　>>>=　&=　^=　¦=</td></tr>
</table>

004 変数について

JavaScriptでの変数の定義

JavaScriptでの変数名は、最初が英文字で始まる必要があります($など一部例外はある)。2文字目以降では、英数字およびアンダーバーを使用することができます。変数名は大文字小文字が区別されます。つまり、「abc」と「Abc」は、別の変数としてみなされます。また、Firefoxなど一部のブラウザを除き、変数名に日本語は使えません。

JavaScriptの変数は宣言することなく、いきなり利用することができます。また、Javaなど他言語と異なり、どんなデータ(数値や文字列、関数、オブジェクト)でも入れることができます。つまり変数の型を問いません。たとえば、「a = 12.3」と変数aに数値の12.3を入れた後で、「a = "OK"」といきなり文字列を入れても問題ありません。また、「a = document」のようにオブジェクトやメソッド、プロパティも入れることができます。

型を問わないので、次のように関数を入れることもできます。

```
a = function(){ alert("OK"); }
```

このようにJavaScriptの変数は、非常に柔軟な仕様となっています。

変数に数値や文字列を代入した場合はそのものが変数に入りますが、オブジェクトなどを指し示す場合には注意が必要です。オブジェクトへの参照を代入するだけでオブジェクトそのものがコピーされ、新たに変数に入るわけではないからです。たとえば、次のサンプルでは変数「a」にページ全体のスタイルシートオブジェクトへの参照を代入しています。その後、変数「b」に変数「a」を代入していますが、この場合はページ全体のスタイルシートオブジェクトが複製され、変数「b」に代入されるわけではありません。単純にオブジェクトへの参照だけが変数「b」に入ります。このため、次のサンプルを実行すると、ページ全体の背景色が赤色になります。「b = a」は「b = document.body.style」と等価になるということです。

```
a = document.body.style;
b = a;
b.backgroundColor = "red";
```

ブラウザに実装されているJavaScriptでは、単純に変数を作成すると「window」オブジェクトのプロパティとして追加されます。つまりグローバル変数という名目になっていますが、実際には「window」オブジェクトのプロパティとして割り当てられます(「for...in」で「window」オブジェクトの中身を確認するとわかる)。このため、「window」オブジェクトのプロパティ名と変数名が重複してしまうと、正常に動作しなくなります。次のサンプルを実行すると、変数「a」の内容も「window」オブジェクトの「a」プロパティの内容も同じ文字が表示されます。

```
a = "OK";
alert(a);
alert(window.a);
```

関数名も同様で、「window」オブジェクト内に割り当てられます。

グローバル変数とローカル変数

JavaScriptでは、グローバル変数とローカル変数があります。グローバル変数は上記のように「window」オブジェクトに割り当てられますが、関数内で「var」宣言した場合のみ、ローカル変数として定義されます。たとえば、次のプログラムの場合、ローカル変数として扱われるのは、変数「d」のみです。他の変数「a」「b」「c」はすべてグローバル変数として割り当てられます。

```
a = 12;
var b = 34;
function test(){
	c = 56;
	var d = 78;
}
```

また、関数内でvar宣言した変数は、その関数内のみ有効になります。ただし、関数内であれば、どこでも参照できてしまうため、注意が必要です。たとえば、次のサンプルのように「for()」などのループブロック内で「var」宣言を行ってもブロック外で変数を参照することができてしまいます。

```
(function(){
	for (var i=0; i<10; i++);
	alert(i);
})();
```

Firefox 2で実装されているJavaScript 1.7では「let」を利用することでブロック内のみ有効な変数として扱うことができますが、他のブラウザでは利用できないので、どうしてもループ外に変数を参照させたくない場合には「(function() {~})()」として関数内に記述する必要があります。ライブラリを制作する場合や、より安全にコードを記述したい場合には有効です。

変数の削除

定義した変数を削除するには、「delete」を使います。ただし、「var」宣言をした変数は削除できないので注意が必要です。また、関数は、変数に入れた場合は削除することができますが、そうでない場合には削除することができません。たとえば、次のサンプルでは変数「c」に定義された関数は削除されますが、関数「d」は「delete」を使っても削除することはできません。

```
c = function (){ alert("OK"); }
delete c;
function d(){ alert("NG"); }
delete d;
d();
```

なお、変数でなく定数として定義するには、「const」を使います。ただし、利用できるブラウザがFirefoxなど限定されます。

005 配列変数/連想配列について

配列変数/連想配列とは

JavaScriptでは、配列変数および連想配列(ハッシュ)を利用することができます。配列変数は複数の値を同時に格納することができる変数で、参照する位置(添え字)を指定して値を呼び出します。連想配列は配列変数と同様に複数の値を同時に格納することができる変数です。通常の配列変数では添え字に数値を利用しますが、連想配列では添え字に文字列を使うことができます。配列変数と連想配列は、JavaScriptではオブジェクトとして処理されます。

配列変数の利用方法

配列変数を作成するには、「Array」オブジェクトを生成することで行います。たとえば、配列変数「a」を作成する場合には、次のように記述します。

```
a = new Array();
```

このときに「Array()」のパラメータに1つだけ数値を指定すると、指定した数の配列変数が作成されます。ただ、JavaScriptでは状況に応じて自動的に配列変数のサイズが調整されるので特に指定しなくても問題ありません。サイズを指定した場合でも、状況によって自動的にサイズは調整されます。
「Array()」のパラメータが複数ある場合には、配列の最初から順番に値が入ります。たとえば、次のように記述すると、配列要素は、1番目が数字の「1」、2番目が数字の「2」、3番目が文字列の「OK」、4番目には文字列の「NG」が入ります。

```
a = new Array(1, 2, "OK", "NG");
```

「new Array()」と記述するのが面倒な場合には、[]を使うこともできます。次のサンプルは、上記サンプルと同じ結果になります。

```
a = [1, 2, "OK", "NG"];
```

JavaScriptの配列は、変数同様に数値でも文字列でもオブジェクトでも関数でも何でも入れることができます(31ページのコラム参照)。
配列の内容を参照/設定するには、[]を使って参照する位置(添字)を指定します。たとえば、1番目の配列の内容を読み出すには、次のように記述します。

```
a = [1, 2, "OK", "NG"];
b = a[0];
```

変数「b」には、配列変数「a」の1番目の内容が代入されます。2番目の内容を読み出すには、「a[1]」となります。配列要素の番号は、「0」から始まる点に注意してください。内容を読み出すのではなく、書き換える場合も同様に指定します。次のサンプルは、配列要素の3番目に「215」の数値を入れます。

```
a[2] = 215;
```

また、配列要素の総数は、「length」プロパティで求めることができます。たとえば、次のサンプルでは、配列変数「a」の要素の総数である「4」が表示されます。

```
a = [1, 2, "OK", "NG"];
alert(a.length);
```

配列内に配列を入れることで、二次元配列や多次元配列を実現することもできます。たとえば、次のように設定/参照することができます。

```
a = [["A","B"],["C","D"],["E", "F"]];
b = a[2][0];
```

この場合、変数「b」には「E」の文字が入ります。

連想配列の利用方法

連想配列(ハッシュ)は、設定時には「{}」を使います。連想配列を定義するには、次のように記述します。

```
a = { year:1969, name:"KF" }
```

連想配列ではキーと値のペアを「,」(カンマ)で区切って列記します。連想配列の値には、変数同様に数値でも文字列でもオブジェクトでも関数でも何でも指定することができます(31ページのコラム参照)。上記の連想配列の設定は、次のように記述することもできます。

```
a = new Object();
a.year = 1969;
a.name = "KF";
```

連想配列へのアクセスは配列変数と同様に「[]」を使います。先ほどの例でキーが「year」の内容を読み出すには、次のようにします。

```
b = a["year"]
```

読み出すキーがわかっている場合にはよいのですが、そうでない場合には「for...in」または「in」を利用します。すべてのキーの値を読み出すには、次のようになります。

```
a = { year : 1969, name : "KF" };
for (key in a){
     alert(a[key]);
}
```

このサンプルでは、変数「key」にキーが入ります。このキーに対応した値を読み出すには、「a[key]」とします。

すべてのキーではなく、特定のキーが存在するかどうかは「if...in」で調べます。次のサンプルは「year」というキーが存在する場合にアラートダイアログを表示します。

```
a = { year : 1969, name : "KF" };
if ("year" in a) alert("プロパティは存在しています");
```

連想配列とオブジェクトの関係

JavaScriptではオブジェクトのプロパティ名と内容は、そのまま連想配列のキーと値になっています。つまり連想配列を作成すれば、それはオブジェクトのプロパティ/メソッドを定義したのと同じことになります。逆にオブジェクトを生成した場合には、オブジェクトのプロパティやメソッドは連想配列として参照することができるわけです。JavaScriptで多く利用される入力フォームの「forms」「elements」などは参照番号と連想配列のキーを対応させてあります。このため、HTML文書の最初の「<form name="eForm1">」にアクセスするには、「document.forms[0]」でも「document.forms["eForm1"]」でも可能になります。ただし、ごく一部のオブジェクトに限られていて、すべての連想配列がこうなっているわけではありません。これは過去との互換性のため、このようになっています(DOM Level 0互換のため)。

Column 配列変数・連想配列内に関数を設定する際の注意点

Safariでは、配列変数内に関数を設定する際に関数名を指定するとエラーになります。たとえば、次のサンプルで上の行はSafariで正しく処理されますが、下の行は文法エラーになります。他のブラウザではエラーなく処理されます。ただし、下の行では関数名に「b」を指定していますが、エラーなく処理されても、実際には「b」という名前の関数は定義されず、無名関数/匿名関数扱いになります。

```
a = [function (){ }];
a = [function b(){ }];
```

◉Safariでエラーになっている画面

JavaScript Console
SyntaxError – Parse error
file:///IDE200GB/DATA/GENKOU/JS%E3%83%86%E3%82%AF%E3%83%8B%E3%83%83%E3%82%AF%E3%83%95%
E3%82%99%E3%83%83%E3%82%AF/PDF/1%E7%AB%A0%E8%BF%8D%E5%8A%A0%E6%96%87%E7%AB%A0/
sample/main.js
Clear
1 message

006 変数の型について

JavaSriptでの変数の型

JavaScriptの変数には、どんなものでも入れることができます。つまり変数に入れるデータの型を問わないことになります。変数の型を問わないのは非常に便利な反面、思わぬ結果を引き起こすことがあります。

型の不一致によるエラーや思わぬ結果を防ぐために、あらかじめ変数の型を調べてから処理する方が安全なことがあります。JavaScriptで変数の型を調べるには「typeof()」を使います。パラメータに調べたい変数名などを指定します。返される方の種類は、下表のようになります。「undefined」は「undefined」、「null」は「object」になる点に注意してください。

typeofが返す文字列	型
number	数値
string	文字列
function	関数
object	オブジェクト／null
undefined	未定義データ／undefined
boolean	真偽値（ブーリアン）

JavaScriptで型が不一致の場合には、自動的に適切な型に変換され処理が行われます。たとえば、「12 + "34"」は数値の「12」が文字列に変換され、「1234」という文字列になります。

型変換を行う方法

JavaScriptで型変換を行う場合には、状況に応じていくつかの方法があります。

▶数値から文字列

数値から文字列に変換する場合は、次のようになります。

```
""+数値
```

```
''+数値
```

```
new String(数値)
```

たとえば、「myVar = ""+123」「myVar = '' + 123」のように記述します。

▶文字列から数値（文字列は数値/数式であること）

文字列から数値に変換する場合は、次のようになります。

```
parseFloat(文字列)
```

```
parseInt(文字列)
```

```
eval(数式を含む文字列)
```

```
new Number(文字列)
```

たとえば、「myVar = parseFloat("1965.215")」「myVar = parseInt("2.15")」「myVar = eval("123+456+Math.PI")」のように記述します。

007 条件分岐について

ifの基本

JavaScriptには、一般的なプログラミング言語同様の条件分岐命令が用意されています。もっとも基本的な命令は「if」命令です。たとえば、変数「a」の値が12かどうかを調べ、値が12の場合にアラートダイアログを表示する場合には、次のように記述します。

```
if (a == 12) alert("12です");
```

「a == 12」でなく、間違って「a = 12」とすると、変数「a」に数値の12が代入されてしまうので注意してください。「if」命令で指定できる比較演算子は、下表のようになります。比較される対象の型が異なる場合には、自動的に適切な型に変換されてから比較が行われます。

比較演算子	意　味
==	式が等しい
===	式が等しく型も等しい
!=	式が等しくない
>	左が大きい
>=	左が等しいか大きい
<	右が大きい
<=	右が等しいか大きい

複雑な条件の指定

複数の条件を満たした場合に処理するには、「&&」を使います。たとえば、変数「a」の値が12で、変数bの値が34の場合にアラートダイアログを表示するには、次のように記述します。

```
a = 12;
b = 34;
if ((a == 12) && (b == 34)) alert("12と34です");
```

どちらか一方の条件を満たした場合に処理するには、「||」を使います。たとえば、変数「a」の値が12か34の場合にアラートダイアログを表示するには、次のように記述します。

```
a = 12;
if ((a == 12) || (a == 34)) alert("12か34です");
```

「&&」と「 || 」を複数使ってより複雑な条件も指定することができます。また、最近では「 || 」は次のように使われることもあります(本書でも一部使用しています)。

```
a = (b || 12);
```

これは変数またはオブジェクト「b」が未定義の場合には変数「a」に12を入れるという内容になります。すでに、オブジェクトとして存在している場合には初期化しないといった場合に利用することができます。

比較演算子を省略した記述

if命令では、比較演算子が省略されて記述されていることがあります。たとえば、次のような記述です。

```
if (a) alert("aは存在しています");
```

これは変数またはオブジェクト「a」が「undefined」または「null」、「false」でない場合(つまり真)に条件を満たしたことになり、以後の命令が実行されます。これとは逆に、変数またはオブジェクト「a」が「undefined」または「null」、「false」の場合に処理するには、否定演算子である「!」を先頭に付けて記述します。

```
if (!a) alert("aは存在していません");
```

条件を満たしたときに複数の命令を実行する記述

「if」命令で条件を満たしたときに実行する命令が複数ある場合には、ブロックを示す「{」と「}」で実行する命令を囲みます。たとえば、次のように記述します。

```
if (a == 12) {
     b = 34;
     alert("aは12です");
}
```

条件を満たした場合、変数「b」に34の値が入り、アラートダイアログが表示されます。

条件を満たした場合とそうでない場合に処理を分ける方法

条件を満たした場合とそうでない場合を分けるには「else」を使い、次のように記述します。条件を満たさなかった場合の処理を「else」以後に記述します。

```
if (a == 12) {
     b = 34;
     alert("aは12です");
}else{
     b = 56;
     alert("aは12ではありません");
}
```

「if」命令の入れ子について

「if」命令は次のように入れ子にすることもできますが、読みにくくなる上にミスも発生しやすくなります。あまりに条件式が長いようであれば考え方を見直すのがよいでしょう。

```
if (a == 12) {
     if (b == 34){
           if (c == 56){

           }
     }
}
```

「switch...case」の使い方

1つの変数の値に応じて複数の処理を行いたい場合には、「if」命令よりも「switch...case」の方が適しています。たとえば、プロンプトダイアログで入力された文字列に応じて処理を振り分けるには、次のように記述します。なお、caseで調べる文字列、値は自動的に型変換が行われないので注意が必要です。

```
a = prompt("英文字を入れてください", "a");
switch(a){
     case "a" : alert("aです");
                              break;
     case "b" : alert("bです");
                              break;
     case "c" : alert("cです");
                              break;
     default : alert("abc以外です");
}
```

Column 条件式を短くする記述方法

JavaScriptでは「switch...case」や「if」命令を列記しなくても連想配列をうまく利用することで条件式を短くすることができます。たとえば、上記の「switch...case」のサンプルは、次のように記述することができます。

```
func = {
     "a":function(){alert("aです")},
     "b":function(){alert("bです")},
     "c":function(){alert("cです")},
};
str = prompt("英文字を入れてください", "a");
if (func[str])
 func[str]();
else
 alert("abc以外です");
```

008 繰り返し処理について

繰り返しを行う命令

JavaScriptには繰り返しを行う命令として「for()」「while()」「do...while()」が用意されています。

一定回数処理を繰り返す

「for()」命令は、C言語など他の言語と同じように記述します。「for()」のパラメータは初期値、終了条件、増減値の3つを指定します。たとえば、0から9までの合計を求めるには、右のように記述することができます。また、初期値、終了条件、増減値は「,」(カンマ)で区切って複数列記することができます。たとえば、下のように記述することができます。

```
n = 0;
for (i=0; i<10; i++) n = n + i;
```

```
n = 0;
m = 0;
for (i=0, j=1; i<10, j<5; i++, j++) { n = n + i; m = m + j; }
```

この場合、「j」の値が5以上になったら終了するので変数「n」の値は6、「m」は10となります。

なお、初期値、終了時条件、増減値とも省略することができます。ただし、次のように全部省略すると無限に繰り返し処理が行われてしまうので注意してください。

```
for (;;)
```

このような場合には、次のように「break」を使って繰り返し処理から抜け出すことができます(直近の繰り返しから抜ける)。また、「continue」で繰り返しの先頭に戻ることができます。

```
n = 0;
for(;;){
    n++;
    if (n>10) break;
}
```

条件を満たしている間は繰り返す

「for()」命令は一定回数繰り返す場合に便利ですが、条件を満たしている間は繰り返すには「while()」または「do...while()」を使います。「while()」のパラメータに条件を指定します。

「while()」の場合は繰り返し処理前に条件が判定されるため、一度も繰り返し処理が行われないことがあります。これに対して「do...while()」では条件は最後に判定されるため、最低でも1回は繰り返し処理が行われます。記述できる条件式は、「if」命令と同じで複数の条件式を組み合わせて指定することができます。

最初に説明した「for()」命令と同じ処理を「while()」を使って記述すると、右のようになります。

```
n = 0;
i = 0;
while (i<10){
    n = n + i;
    i++;
}
```

SECTION

009 関数の定義

関数の定義

JavaScriptで関数を定義するには、「function」または「new Function()」を使って行います。一般的には、「function」を使います。たとえば、「abc」という名前の関数を定義するには、次のように記述します。

```
function abc(){ }
```

JavaScriptでは、変数に関数も入れることができるため、次のように記述しても同じです。

```
abc = function(){ }
```

また、関数の名前は付けなくても動作させることができます（無名関数／匿名関数）。たとえば、次のように記述することができます。

```
(function(){ alert("Ok"); })()
```

「function」でなく「new Function()」の場合にはパラメータに実行する関数の内容を指定します。たとえば、次のように記述することができます。

```
test = new Function("alert('ok')");
test();
```

関数名を指定した場合の扱い

「function」で単純に関数名を指定した場合には、「window」オブジェクトのメソッドとして定義されます。次のサンプルでは、アラートダイアログが2回表示されます。

```
function abc(){ alert(123); }
abc();
window.abc();
```

関数を定義する際の注意点

JavaScriptでは同じ名前の関数があれば、後から定義した方が有効になります。また、JavaScriptではオブジェクトのひな形を定義する場合やメソッドの定義にも「function」を使います（38ページ参照）。そのため、間違ってすでにある「window」オブジェクトのメソッドやプロパティを書き換えないように注意してください。

010 オブジェクト/メソッド/プロパティについて

オブジェクト

JavaScriptで、さまざまな処理を行う際に利用されるのが「オブジェクト」です。JavaScriptでは、HTML文書もスタイルシートもオブジェクトとして扱われます。ページ上に表示されているものは、すべてオブジェクトと考えても構いません。また、三角関数や対数などの計算を行うオブジェクトや文字列操作を行うオブジェクトなど、さまざまなオブジェクトが用意されています。

オブジェクトでもっとも基本となるのが「Object」オブジェクトです。JavaScriptでは既存のオブジェクトを利用するだけでなく、「Object」を使って独自にオブジェクトを定義することができます。たとえば、「computer」という名前のオブジェクト（コンストラクタ）を用意し、そこからさまざまなオブジェクト（インスタンス）を生成するには、次のように記述します。

```
function computer(machineName, cpu){
     this.name = machineName;
     this.cpu = cpu;
     this.memory = "1GB";
}
macBook1 = new computer("MacBook", "Core2 Duo");
```

オブジェクトを生成するには、「new」を使って行います。上記では、「new」を使って「macBook1」という名前のオブジェクト（インスタンス）が生成されます。

「function」は関数を定義する命令ですが、JavaScriptではオブジェクトのひな形（コンストラクタ）を定義する場合にも使用します。基本的には関数の定義と変わりませんが、オブジェクトのプロパティやメソッドを定義する場合には、「this」を使ってオブジェクト自身を示します。

プロパティ

オブジェクトのプロパティは「オブジェクト名.プロパティ名」として設定/参照することができます。プロパティの定義は上記サンプルのように「this.プロパティ名」として行うことができます。

メソッド

JavaScriptでは、プロパティに実行するプログラムコードを指定したのが「メソッド」として処理されます。次のサンプルでは「play」メソッドを定義し、メソッドが呼ばれたらアラートダイアログを表示します。

```
function computer(machineName, cpu){
     this.name = machineName;
     this.cpu = cpu;
     this.memory = "1GB";
     this.play = function(){ alert(this.name+"で音楽を再生します");}
```

```
}
macBook1 = new computer("MacBook", "Core2 Duo");
macBook1.play();
```

プロパティ/メソッドの上書き

JavaScriptでは後からプロパティ/メソッドの内容を上書きすることできるので、基本となるオブジェクトを定義し、状況に応じて追加していくことができます。これは自分で作成したオブジェクトだけでなく、既存のオブジェクトでも同様です。たとえば、文字列の先頭に★マークを付加するメソッドを追加するプログラムは次のようになります。

```
String.prototype.star = function(){
    return "★"+this;
}
alert("Sample".star());
```

実行すると「★Sample」と表示されます。既存のオブジェクトにプロパティやメソッドを追加する場合には「prototype」を使って行います。「オブジェクト名.prototype.メソッド名 = function() { 処理 }」として好きなだけ機能を追加することができます。JavaScriptライブラリでもっとも多く利用されているprototype.jsライブラリは、この機能を利用してさまざまなメソッドを既存のオブジェクトに追加してJavaScriptの基本機能を強化しています。

Column whileでプロパティなどの状態の変化をチェックしない

JavaScriptには条件を満たしている間に処理を行う「while()」(36ページ参照)がありますが、プロパティの変化などを「while()」でチェックしてはいけません。場合によってはブラウザが停止してしまいます。定期的に何かプロパティなどを監視したりチェックする場合には、「while()」ではなく、「setTimeout()」(64ページ参照)か「setInterval()」(66ページ参照)を使って行うのが安全です。

また、状態が変化したときに発生するイベントが用意されているのであれば、そのイベントに対してイベントハンドラ(処理)を設定します。

011 イベントについて

イベントの概要

JavaScriptでは、ユーザーの操作やブラウザからの応答によってさまざまなイベントが発生します。このイベントは、ブラウザによって実装が異なります。つまり、Internet Explorerでは存在するイベントがFirefoxでは存在しないといったことがあります。もちろん、基本的なイベントに関しては、どのブラウザでも捕捉することができます。

しかし、イベントを扱うイベントオブジェクトに関しては、ブラウザによって対応が異なります。大きく分けると、Internet ExplorerとFirefoxなど標準仕様に沿ったブラウザになります。問題なのは、これらのブラウザで処理も命令もプロパティ名さえも異なっており、互換性も少ないという点です。このため、イベント処理に関してはブラウザごと対処しなければならないことになります。なお、SafariやOperaの場合には、Internet Explorer用のプログラムでもFirefox用のプログラムでも動作するようになっています。

イベントの記述方法

イベントの設定方法も旧来の方法を除くと、Internet ExplorerとFirefoxなど標準仕様に沿ったブラウザで異なります。旧来の方法であれば、イベントハンドラの定義に関してはどのブラウザでも同じように扱うことができます。たとえば、旧来の方法で<a>タグがクリックされたときにアラートダイアログを表示するには、次のように記述します。

```
<a href="#" onClick="alert('Ok')">クリックしてください</a>
```

上記の記述方法は、古いブラウザでも新しいブラウザでも確実に動作します。どうしても、動作保証を取らなければならない場合には、これでも構わないでしょう。

しかし、2007年現在ではHTML文書とスタイルシートファイル、JavaScriptコードは分離する方向です。この場合、次のように<a>タグにID名を指定し、ページが読み込まれたらonclickイベントハンドラを設定します。

SOURCE CODE HTML

```
<a href="#" id="aLink">クリックしてください</a>
```

SOURCE CODE JavaScript

```
window.onload = function(){
  document.getElementById("aLink").onclick = function(){
    alert("ok");
  }
}
```

通常は、この方法でも問題ありませんが、開発者が複数人に渡る場合やライブラリを使用している場合には、不具合が発生することがあります。それは、「onclick」などのイベントハンドラに上記のように処理を設定すると、以前に設定されていた処理が上書きされてしまい、実行されなくなるためです。たとえば、次のサンプルではアラートダイアログを2回表示するのを期待していますが、最後に定義された処理しか動作しないため、アラートダイアログは「ok」を表示するだけです。

```
window.onload = function(){
  document.getElementById("aLink").onclick = function(){
    alert("NG");
  }
  document.getElementById("aLink").onclick = function(){
    alert("ok");
  }
}
```

サンプルでは見ればすぐにわかりますが、イベント設定部分が離れている場合には、このような不具合は発見が難しくなります。そこで、現在では、「addEventListener()」を使って、イベントを上書きするのではなく、追加するようにします。ただ、「addEventListener()」はInternet Explorerでは使用できないため、代わりとなる「attachEvent()」を使って行います。

上記サンプルを「addEventListener()」「attachEvent()」を使って書き換えると、次のようになります。

SOURCE CODE HTML

```
<a href="#" id="aLink">クリックしてください</a>
```

SOURCE CODE JavaScript

```
window.onload = function(){
  link1 = document.getElementById("aLink");
  if (link1.addEventListener) link1.addEventListener("click", ⏎
  function(){ alert("NG");}, false);
  if (link1.attachEvent) link1.attachEvent("onclick", function(){ ⏎
  alert("NG");});

  if (link1.addEventListener) link1.addEventListener("click", ⏎
  function(){ alert("ok");}, false);
  if (link1.attachEvent) link1.attachEvent("onclick", function(){ ⏎
  alert("ok");});
}
```

上記のサンプルでは、リンク文字をクリックすると、アラートダイアログが2回表示されます。なお、「addEventListener()」「attachEvent()」で追加したイベントを削除するには、「removeEventListener()」「detachEvent()」を使います。パラメータには、追加時に指定したパラメータと同じパラメータを指定します。パラメータが異なると違うイベントとみなされ削除されません。

イベントオブジェクトについて

イベントが発生すると、イベントオブジェクトにイベントが発生したエレメントやX/Y座標値など各種情報が設定されます。ただし、このイベントオブジェクトの扱いがブラウザにより異なっています。

Internet Explorerでは「event」オブジェクトに情報が格納され、Firefoxではイベント処理を行う関数（イベントハンドラ）のパラメータとして渡されます。Safari2やOperaでは、「event」オブジェクトと、イベント処理を行う関数（イベントハンドラ）のパラメータの両方に情報が格納されます。次のサンプルを実行すると、「event」オブジェクトに格納されたか、パラメータとして渡されたかを確認することができます。

SOURCE CODE　HTML

```
<a href="#" id="aLink">クリックしてください</a>
```

SOURCE CODE　JavaScript

```
window.onload = function(){
  document.getElementById("aLink").onclick = function(evt){
    alert(evt);
    alert(event);
  }
}
```

大きく分けると、Internet ExplorerとFirefoxなど標準仕様に沿ったブラウザで処理分けることになります。座標値の取得やエレメントの取得なども異なっているため、イベント処理に関してはブラウザ別に対処する必要があります。

ブラウザ別にイベント処理を分けるには以下のようにします。最初に関数にイベントオブジェクトが渡されたかどうかを判別することでInternet Explorerだけ処理を分ける事ができます。

```
  function eventProc(evt){
    if (!evt) {
      【Internet Explorerでのイベント処理】
    }else{
      【DOMに準拠したブラウザでのイベント処理】
      (Firefox, Safari, Operaなど)
    }
  }
```

012 DOMについて

DOMの概要

JavaScriptでは、HTML文書をオブジェクトとして扱うことができます。これはHTMLのタグやテキストや画像など、ほぼすべてがオブジェクトとして処理されます。これをDOM(Document Object Model/ドキュメントオブジェクトモデル)といい、HTMLタグもオブジェクトなのでプロパティやメソッドによって操作したり、各種処理を行ったりすることができます。

HTMLタグだけでなく、スタイルシートも同様にオブジェクトとして扱われます。たとえば、次のように記述するとスタイルシートの「background-color」プロパティに黄色が設定されページ全体の背景色が黄色になります。

```
document.body.style.backgroundColor = "yellow";
```

記述方法

JavaScriptでページ上のオブジェクトにアクセスする場合には、注意が必要です。特に以前から指定されている方法と新しい方法では記述方法が異なるためです。以前は以下のように参照番号やハッシュ、オブジェクトのプロパティ名として指定されていました。

```
  <img src="images/01.jpg">
  <img src="images/02.jpg" name="im2">
  <img src="images/03.jpg" name="im3">

document.images[0].src = "images/ok.jpg";
document.im2.src = "images/ok.jpg";
document.images["im3"].src = "images/ok.jpg";
```

もちろん、最新のブラウザでも古いブラウザでも問題なく動作します。特に古いブラウザや多くのブラウザに対応させなければならない場合は、このような書き方の方がよいこともあります。

新しい書き方はDOM仕様に沿って、タグやテキストをノードとして扱います。ノードに対して各種操作を行うことで、テキストを追加したり画像を変更します。上記の画像を入れ替えるサンプルを書き換えると、次のようになります。

```
  <div id="photo">
    <img src="images/01.jpg">
    <img src="images/02.jpg">
    <img src="images/03.jpg">
  </div>
```

```
var photoArea = document.getElementById("photo");
var img = photoArea.getElementsByTagName("img");
for (var i=0; i<img.length; i++){
  img.item(i).src = "images/ok.jpg";
}
```

基本的には操作対象とするノードにID名を指定し、「getElementById()」を使って情報を取得します。ID名だけでなく、スタイルシートのクラス名や特定の属性が利用されることもあります。サンプルのように複数の画像を操作する場合には、「getElementsByTagName()」を使って特定のタグのノードリストを取得します。ノードの参照は「item()」で行うことができますが、HTML文書の場合、「img[i]」のように配列形式で指定しても動作します。XMLなどを操作する場合には、「item()」を使うようにしてください。本書ではHTML文書のみ扱っているので「img[i]」のように配列として処理を行っています。

Column JavaScriptコードをコンパクト化する

開発が大規模なものになるとJavaScriptコードが肥大化し読み込みまでに時間がかかります。何万人ものユーザーがアクセスする場合にはプログラムコードのサイズにも注意しなければなりません。幸いにしてJavaScriptコードをコンパクト化するツールがいくつか出ています。その1つが次のURLにある「jsmin」です。

URL http://www.crockford.com/javascript/jsmin.html

このツールはJavaScriptの不要な改行コードや空白だけをカットして詰め込みコードをコンパクト化します。使用方法はコマンドラインから以下のようにコンパクト化していないファイル名（fullsize.js）とコンパクト化した際のファイル名(compactSize.js)を指定します。

```
./jsmin.rb <./fullsize.js >compactSize.js
```

SECTION 013

KNOWLEDGE

プログラムを作成する場合の注意点

プログラムを作成する場合の注意点

近年、JavaやC/C++言語などからJavaScriptを扱うプログラマが増加したせいか、これまでとは異なるスタイルのプログラムを見かけることもあります。プログラム言語は多くの種類があり、関数型言語、オブジェクト指向型言語など大きなカテゴリで分けられることもできます。

JavaScriptはオブジェクト指向型の言語ですが、非常に柔軟性に富んでいるため同じ動作をするプログラムを記述する場合でも、関わってきた言語によってスタイルが変わってしまいます。特にJavaScriptライブラリなどは、オブジェクト指向をより強めたprototype.jsのようなライブラリもあれば、関数指向のMochiKitなどもあります。

プログラムスタイルは人それぞれですが、さまざまなブラウザ上で動くJavaScriptコードを記述するのは意外と面倒です。ライブラリを利用すれば、ある程度は吸収してくれますが、それでも限度があります。Firefoxで動いてもInternet ExplorerやSafariで動かないことは多々あります。たとえば、XMLデータやHTMLデータでノードを参照する場合でも、しっかりとした互換性がなく、異なるブラウザで動作させるのは手間がかかります。この辺りは、やはりWebサイトを参照したり、ノウハウをためるしかありません。

数年前に書かれたコードが、美しくないとする新規参入プログラマも多いでしょう。しかし、現実問題として、そのように書かなければ動作しないことがあったともいえます。1行にまとめて記述すればよいのに2行に分けないと動作しないというようなこともありますし、メソッドからの戻り値が異なっているなど仕様の違いも多くあります。おまけに、新しいブラウザが出ると、そのブラウザの不具合にも対応しなければなりません。

Webページ上で使用されるJavaScriptプログラムの寿命は長くないため、とりあえず動けばよいという考え方もあります。ただ、今後長く使えるような技術であるXMLやDOMのノウハウは習得しておいて損はないでしょう。また、自分が書いたコードが他人に引き継がれるときのことも考慮して作成しておくと、よりよいものになるはずです。

Column 名前空間とは

名前空間とは、複数のライブラリを利用する際などに、変数名やオブジェクト名などが重複してしまい、問題が起きることを避けるための概念です。JavaScriptには、名前空間を指定するための命令が用意されていません。このため、ライブラリなどを利用した場合などに変数名やオブジェクト名が衝突してしまうことがあります。

そこで、JavaScriptでは、擬似的に名前空間を利用する方法がとられています。基本的にはオブジェクトを生成し、そのプロパティ名を名前空間として利用します。また、Yahoo UIライブラリでは名前空間を指定するためのメソッドが用意されています。たとえば、次のサンプルでは「KF」という名前の名前空間を指定しています。

```
YAHOO.namespace ("KF");
YAHOO.KF = function(){ alert("名前空間サンプル");}
YAHOO.KF();
```

014 JavaScriptライブラリについて

JavaScriptライブラリの種類

Ajaxの普及とともに、多くのJavaScriptライブラリが登場しています。JavaScriptのライブラリには、次のような種類があります。

▶ prototype.js

JavaScriptライブラリの中でもっとも多く利用されているのが、prototype.jsライブラリです。このライブラリはJavaScriptの基本的なオブジェクトを拡張することにより、多くの機能を提供してくれます。また、よりオブジェクト指向を強めた作りになっています。prototype.jsはJavaScriptの基本オブジェクトを拡張しているため場合によっては不具合を起こすこともあります。オブジェクトを拡張しないライブラリがよいのであれば後述するDojoやYahoo UIライブラリを利用してください。

● prototype.js

URL http://www.prototypejs.org/

▶ script.aculo.us

prototype.jsライブラリとともに多く利用されるのが、主にエフェクト処理やコントロール処理を行うためのscript.aculo.usライブラリです。速度は速くありませんが、さまざまなエフェクトやコントロールを手軽に扱うことができます。

● script.aculo.us

URL http:///script.aculo.us/

▶ Dojo/Yahoo UI Library

より大規模なライブラリとして提供されているのが、DojoとYahoo UIライブラリです。ただし、Dojoは本書執筆時点（2007年4月）では、バージョン1未満であり完成形として提供されていません。このため、バージョンアップされた場合には本書のサンプルが動作しなくなる可能性もあります。

DojoもYahoo UIライブラリも非常に多機能ですが、その分だけライブラリのファイルサイズが大きくなり、読み込みに時間がかかることがあります。Dojoは動的にファイルを読み込みますが、Yahoo UIライブラリの場合には必要最低限のファイルだけを読み込むようにしておきます。また、Yahoo UIライブラリでは名前空間の設定など、大規模なプログラムに配慮した作りになっています。

● Dojo

URL http://dojotoolkit.org/

● Yahoo UI Library

URL http://developer.yahoo.com/yui/index.html

jQuery

jQueryはコンパクトなサイズでありながら、高機能な処理を行うことができるライブラリです。作成されるプログラムもコンパクトになる特徴があります。XML/HTML文書の階層構造などを指定してアクセスすることができたり、標準で主要なエフェクトが利用できます。他のライブラリと異なり、繰り返して定義したり、処理したりする記述が大幅にカットされるため、複雑な処理も短いコードで作成することができます。

● jQuery

URL http://jquery.com/

Adobe Spry

アドビシステムズが提供しているのがAdobe Spryフレームワークです。ライブラリではなくフレームワークとなっているのは、HTML文書内に特定の属性を指定するだけでXMLデータ処理やメニュー、タブパネル、エフェクト処理などを行うことができるためです。ある程度、規格が決まってしまっているため、それに沿ってHTML文書を作成すればよいことになります。このため、他のライブラリと比較して手軽にXMLデータ処理などを行うことができます。

● Adobe Spry

URL http://labs.adobe.com/technologies/spry/

その他のライブラリ

プログラム技術を必要としないライブラリもあります。本書でも使用していますが、スライド表示処理などを行うライブラリがいくつも用意されています。有名なものとしては、「Lightbox」「ThickBox」「GreyBox」「Highslide」などがあります。これらはHTML文書内のタグに特定の属性と値を設定するだけで自動的に処理を行います。

● Lightbox

URL http://www.huddletogether.com/projects/lightbox2/

● ThickBox

URL http://jquery.com/demo/thickbox/

● GreyBox

URL http://orangoo.com/labs/GreyBox/

● Highslide

URL http://vikjavev.no/highslide/

ライブラリの利用方法

ライブラリを利用するにはライブラリが公開されているページでライブラリ一式をダウンロードします。単純にソースコードだけが公開されているのであれば、ローカルディスク上にJavaScriptファイルとして保存します。Yahoo UIライブラリなどはスタイルシートや画像データなども正しいディレクトリ階層に指定しないと動作しない点には注意が必要です。

Break Time

凡ミスほど見つけにくい

プログラムでは、意外と簡単なミスほど見つけにくいものです。半角空白がなかったり、不要な部分に「,」や「;」が記述されている場合は、ミスが見つけにくくなります。また、使用しているブラウザによってもミスが発見しにくくなります。

たとえば、次のサンプルは余計な「,」(カンマ)が末尾に入っています。これは、Internet ExplorerやSafari、Operaではエラーになります。しかし、Firefoxではエラーを起こすことなく、正しく処理されてしまいます。

開発の際には、必ず複数のブラウザで動作を確認することを忘れないようにしてください。その方がミスも発見しやすくなります。

```
a = {
  "type":12,
  test : function(){ alert("test"); },
}
a.test();
```


Firefoxではエラーにならずに処理される

CHAPTER - 02

基本テクニック

015 JavaScriptをHTML文書内に記述する

ここでは、HTML文書内にJavaScriptのコードを記述する方法について解説します。

JavaScript Sample - Windows Internet Explorer
C:¥JavaScript-Sa
Live Search
JavaScript Sample
下の画像はHTML内に記述したJavaScriptで表示しています。

ページ内にJavaScriptで書き出された画像が表示される

SOURCE CODE　HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; ↩
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css" ↩
    media="all">
  </head>
  <body>
    下の画像はHTML内に記述したJavaScriptで表示しています。<br>
    <script type="text/javascript"><!-- ―1
    document.write("<img src='images/DSC0001.jpg'>"); ―2
    // --></script>
  </body>
</html>
```

1 …<script type="text/javascript">として以下のコードがJavaScriptのプログラムであることを示す
2 …JavaScriptのコードを記述する

OnePoint JavaScriptのコードは<script>タグで囲まれた範囲に記述する

HTML文書内にJavaScriptのプログラムを記述するには、<script>タグを使います。<script>タグで囲まれた範囲のプログラムが実行されます。<script>は<head>〜</head>、<body>〜</body>のどちらでも記述することができます。また、<script>タグは1つのページにいくつでも記述することができます。

現在多くのブラウザでは「<script type="text/javascript">」の表記に対応していますが、古いブラウザで問題が発生する場合には「<script language="JavaScript">」とするか、「<script type="text/javascript" language="JavaScript">」のように「type」属性と「language」属性を併記してください。

016 JavaScriptをHTMLタグ内に記述する

ここでは、HTMLタグ内にJavaScriptのコードを記述する方法について解説します。

SOURCE CODE　HTML ▶ index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
  </head>
  <body>
    イベントハンドラを利用してHTMLタグ内にJavaScriptを記述しています。<br>
    ※マウスカーソルを重ねると画像が入れ替わります。<br>
      <img src="images/DSC0001.jpg"
        onmouseover="this.src='images/DSC0002.jpg'" ― 1
        onmouseout="this.src='images/DSC0001.jpg'"> ― 2
  </body>
</html>
```

1 …マウスが重なったときに発生するonmouseoverイベント時の処理を指定する

2 …マウスが離れたときに発生するonmouseoutイベント時の処理を指定する

OnePoint タグ内に記述する場合はイベントハンドラを使う

HTMLタグ内にJavaScriptのプログラムを記述するには、イベントハンドラを利用します。イベントハンドラには、イベント発生時に呼び出される処理を記述します。タグ内で捕捉できるイベントは数多くあります。イベントハンドラは「on～」で始まり、その後にイベント名を記述します。実行するJavaScriptのプログラムは、「=」に続けて記述します。処理が長くなる場合には、あらかじめ処理する内容を関数として定義しておき呼び出すようにします。

注意点としては戻り値が「true」か「false」かによって動作が変わるイベントハンドラがあります。たとえば、<form>タグの「onsubmit」や「onreset」、<a>タグの「onmousover」などがあります。なお、イベントハンドラはHTMLタグ内に記述する場合は、大文字・小文字を問いません(プログラム内では全部小文字にする必要がある)。

SECTION 017

別ファイルのJavaScriptファイルを読み込む

ここでは、別ファイルのJavaScriptファイルを読み込む方法について解説します。

SOURCE CODE　HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; ↩
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css" ↩
    media="all">
    <script type="text/javascript" src="main.js"></script> ―1
  </head>
  <body>
    下のロールオーバーは別ファイルのJavaScriptファイルで作成しています。<br>
      <img src="images/01.jpg" id="photo1">
  </body>
</html>
```

1 …「src」属性で読み込むJavaScriptファイルのURLを指定する

SOURCE CODE　JavaScript main.js

```
window.onload = function(){ ―1
  document.getElementById("photo1").onmouseover = function(){
    this.src = "images/02.jpg";
  }
  document.getElementById("photo1").onmouseout = function(){
    this.src = "images/01.jpg";
  }
}
```

1 …ページが読み込まれたら画像を入れ替えるイベントハンドラを設定する

OnePoint 読み込むJavaScriptファイルは「src」属性でURLを指定する

JavaScriptプログラムが書かれたファイルがHTML文書ファイルとは別になっている場合、<script>タグを使って読み込ませます。読み込むファイルのURLは、「src」属性で指定します。<script>タグ1つにつき1ファイルしか読み込むことができないため、複数のファイルを読み込ませる場合には<script>タグを列記します。なお、<script>タグを複数記述することができない場合は、55ページのようにダイナミックにJavaScriptファイルを読み込ませるようにすることもできます。

018 ダイナミックに別ファイルのJavaScriptファイルを読み込む

ここでは、ダイナミックに別ファイルのJavaScriptファイルを読み込む方法について解説します。

JavaScript Sample - Windows Internet Explorer
動作を確認 sub.jsを読み込み
結果：

最初に［動作を確認］ボタンをクリックすると…

こんにちは

「こんにちは」と表示される

こんにちは

［sub.jsを読み込み］ボタンをクリックすると別のJavaScriptファイルが読み込まれ、再度［動作を確認］ボタンをクリックすると…

今日もいい天気ですね！

読み込まれたプログラムを実行することができる（「今日もいい天気ですね!」と表示される）

SOURCE CODE　HTML ▶ index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script> ―1
  </head>
  <body>
```

```
    <form action="./loadjs.cgi" method="get" name="mainForm">
      <input type="button" id="checkButton" value="動作を確認">
      <input type="button" id="loadButton" value="sub.jsを読み込み">
    </form>
    <div id="result">結果:</div>
  </body>
</html>
```

1 …他のスクリプトを読み込むためのJavaScriptファイルを指定する

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("checkButton").onclick = function(){
    document.getElementById("result").innerHTML = "こんにちは";
  }
  document.getElementById("loadButton").onclick = function(){
    var script = document.createElement("script"); ─1
    script.charset = "utf-8";
    script.src = "sub.js"; ─2
    document.body.appendChild(script); ─3
  }
}
```

1 …<script>タグを生成する
2 …読み込むJavaScriptファイルのURLを指定する
3 …ページ末尾に追加しJavaScriptファイルを読み込ませる

SOURCE CODE JavaScript sub.js

```
document.getElementById("checkButton").onclick = function(){
  document.getElementById("result").innerHTML = "今日もいい天気ですね!";
}
```

OnePoint <script>タグを生成してページに連結する

状況に応じて必要なJavaScriptファイルを読み込ませることができます。まず、「document.createElement()」で<script>タグを生成し、各プロパティを設定します。「src」プロパティには、読み込むJavaScriptファイルのURLを指定します。設定が終わったら「document.body.appendChild()」でページ末尾に追加します。

なお、JSON形式のデータも同様の方法で読み込ませることができます。

019 後付けでイベントを定義する

ここでは、HTML文書のタグに後付けでイベントを定義する方法について解説します。

JavaScript Sample - Windows Internet Explorer
ボタン1 ボタン2
結果：

結果：イベント発生(ボタン1)

結果：イベント発生(ボタン1)
イベント発生(ボタン2)

ボタンをクリックすると、後付けで設定されたプログラムが実行される

SOURCE CODE　HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <form action="./setevent.cgi" method="get" name="mainForm">
      <input type="button" id="checkButton1" value="ボタン1">
      <input type="button" id="checkButton2" value="ボタン2">
    </form>
    <div id="result">結果:</div>
  </body>
</html>
```

1 …イベントを割り当てたいタグにID名を指定する

SOURCE CODE JavaScript main.js

```
if (window.addEventListener) {
  window.addEventListener("load", setEvent1, false);
  window.addEventListener("load", setEvent2, false);
}
if (window.attachEvent) {
  window.attachEvent("onload", setEvent1);
  window.attachEvent("onload", setEvent2);
}
function setEvent1(){
  document.getElementById("checkButton1").onclick = function(){
    document.getElementById("result").innerHTML +=
    "イベント発生(ボタン1)<br>";
  }
}
function setEvent2(){
  document.getElementById("checkButton2").onclick = function(){
    document.getElementById("result").innerHTML +=
    "イベント発生(ボタン2)<br>";
  }
}
```

1 …DOM(Document Object Model)に準拠したブラウザの場合は「window.addEventListener()」でイベントを設定する

2 …Internet Explorerの場合は「window.attachEvent()」でイベントを設定する

OnePoint 後付けでイベントを設定するには「addEventListener」「attachEvent」を利用する

HTML文書内に定義されたボタンなどにイベントを設定するには、あらかじめ設定したいタグにID名を指定しておきます(prototype.jsやjQueryなどのライブラリを利用すればID名を指定しなくても設定することができる)。DOM(Document Object Model)に準拠したブラウザの場合は「addEventListener()」を使ってイベントを追加します。イベント名は「on~」が付かない状態での記述になり、たとえば、「onmouseover」であれば「mouseover」と指定します。

Internet Explorerの場合は「addEventListener()」が使えないので、「attachEvent()」を使ってイベントを追加することができます。

イベントを設定する場合、これらのメソッドではなくオブジェクトにイベントハンドラを指定する形でも可能です。たとえば、本書のサンプルではページの読み込みが完了した場合には、「window.onload=function(){~}」として設定することもできます。ただし、他の場所で同様のコードを記述した場合、後から指定した処理が有効になり、以前の処理は上書きされ消えてしまいます。これを防ぐためには、「addEventListener()」「attachEvent()」を使います。これらのメソッドでは、以前に設定された処理は消されずに追加される形になります。共同作業や複数のライブラリやプログラムを利用する場合には、「addEventListener()」「attachEvent()」を利用する方が安全です。

「addEventListener()」「attachEvent()」で設定したイベントは、それぞれ、「removeEventListener()」「detachEvent()」を使って指定したイベントのみ削除することができます。

020 使用しているブラウザでDOMが使えるかどうか判別する

ここでは、DOM(Document Object Model)が使えるブラウザかどうか判別する方法について解説します。

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    お使いのブラウザでは<br>
    <div id="result">結果:</div> ―1
  </body>
</html>
```

1 …結果を表示する領域を指定する

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  if (document.getElementById){ ―1
    document.getElementById("result").innerHTML = "DOMが利用できます";
  }else{
    document.getElementById("result").innerHTML = "DOMは使えません"; ―2
  }
}
```

1 …DOMに準拠したブラウザの場合は「document.getElementById()」が使えるのを利用して判別する
2 …「document.getElementById()」が使えない場合にはDOMに対応していないとみなす

OnePoint DOMが利用できるかどうかは「document.getElementById」の有無でチェックする

ブラウザがDOM関連のメソッドに対応しているかどうかは、「document.getElementById()」が使えるかどうかで判別を行います。ただし、DOMが使えるブラウザであっても、すべてのDOM関連のメソッドやプロパティを扱うことができるわけではないので注意してください。また、メソッドが利用できてもブラウザによって動作が異なることがあるため、利用の際には注意が必要です。

SECTION 021

使用しているブラウザが何なのかを判別する

ここでは、ブラウザとOSの種類を判別する方法について解説します。

JavaScript Sample - Windows Internet Explorer

ブラウザの判別

ブラウザチェック

結果：

ボタンをクリックすると…

ブラウザの判別

ブラウザチェック

Internet Explorer

Windows

使用しているブラウザとOS名が表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>ブラウザの判別</h1>
    <form action="./chkbrowser.cgi" method="get" name="mainForm">
      <input type="button" id="checkButton" value="ブラウザチェック">
    </form>
    <div id="result">結果:</div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("checkButton").onclick = function(){
    document.getElementById("result").innerHTML = ⤵
    browser.type()+"<br>";
    document.getElementById("result").innerHTML += browser.OS();
  }
}
var browser = {
  type : function(){
    if (window.opera) return "Opera"; ―1
    if (window.createPopup) return "Internet Explorer"; ―2
    if (navigator.userAgent.indexOf("Safari") > -1) return "Safari"; ―3
    if (navigator.userAgent.indexOf("Firefox") > -1) ⤵
    return "Firefox"; ―4
    return "不明"; ―5
  },
  OS : function(){
    if (navigator.userAgent.indexOf("Windows") > -1) ⤵
    return "Windows"; ―6
    if (navigator.userAgent.indexOf("Mac") > -1) return "MacOS"; ―7
    if (navigator.userAgent.indexOf("X11") > -1) return "UNIX"; ―8
  }
}
```

1 …Operaかどうかは「window.opera」プロパティの有無で判別する

2 …Internet Explorerかどうかは「window.createPopup」メソッドが利用できるかどうかで判別する

3 …Safariかどうかはユーザーエージェントの文字列に「Safari」の文字が含まれるかどうかで判別する

4 …Firefoxかどうかはユーザーエージェントの文字列に「Firefox」の文字が含まれるかどうかで判別する

5 …判別不能な場合は「不明」の文字列を返す

6 …OSがWindowsかどうかはユーザーエージェントの文字列に「Windows」の文字が含まれるかどうかで判別する

7 …OSがMacOSかどうかはユーザーエージェントの文字列に「Mac」の文字が含まれるかどうかで判別する

8 …OSがUNIX(互換OS含む)かどうかはユーザーエージェントの文字列に「X11」の文字が含まれるかどうかで判別する

OnePoint ブラウザの判別は独自プロパティ/メソッドとユーザーエージェントを利用する

ブラウザが独自のプロパティやメソッドを持っているかどうかでブラウザの判別を行います。その後でユーザーエージェントの文字列内にブラウザ固有の文字列が含まれているかどうか調べます。ただし、ユーザーエージェントは必ずしも使用しているブラウザを示しているとは限らないため100%ブラウザを判別できない場合があります。

また、OS名を調べるには、ユーザーエージェント内にOSを示す文字列があるかどうかで判別を行います。OSの判別もブラウザの判別同様にユーザーエージェントが正しいとは限らないため正確に判別できない場合があります。

SECTION

022 オブジェクトや変数の型を調べる

ここでは、オブジェクトの型を調べ、それぞれの型に応じて処理を振り分ける方法について解説します。

SOURCE CODE　HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; 
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css" 
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h3>数値、文字列、関数でそれぞれ処理を振り分けています。</h3>
    <div id="result">結果:</div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  var res = document.getElementById("result");
  res.innerHTML = checkType(123) +"<br>"; ―1
  res.innerHTML += checkType("Jinbouchou") +"<br>"; ―2
  res.innerHTML += checkType(jsTest) +"<br>"; ―3
}
function checkType(checkData){
  if (typeof(checkData) == "number") return checkData * 2; ―4
  if (typeof(checkData) == "string") return checkData.length; ―5
  if (typeof(checkData) == "function") return checkData(); ―6
}
function jsTest(){
  return "こんにちは";
}
```

1 …引数に「123」という数値を指定して関数「checkType」を呼び出す
2 …引数に「Jinbouchou」という文字列を指定して関数「checkType」を呼び出す
3 …引数に「jsTest」という関数を指定して関数「checkType」を呼び出す
4 …「typeof()」の結果が「number」であれば型は数値で、2倍にして返す
5 …「typeof()」の結果が「string」であれば型は文字列で、文字列の長さを返す
6 …「typeof()」の結果が「function」であれば型は関数で、関数を実行し結果を返す

OnePoint オブジェクトの型は「typeof()」で調べる

JavaScriptでは変数やオブジェクトの型は、状況に応じて変化し、固定的ではありません。自由度が高く便利な反面、数値と思って加算したら文字列として連結されてしまったという状況もあります。このような場合は、「typeof()」を使って型を調べて処理を振り分けます。

「typeof()」の戻り値	内　容
boolean	ブーリアン（真偽）
function	関数
number	数値
object	オブジェクト
string	文字列
undefined	未定義

023 オブジェクトが存在しない場合のみ新たにオブジェクトを生成する

ここでは、オブジェクトが存在しないときにボタンをクリックした場合のみ、新たにオブジェクトを生成する方法について解説します。

オブジェクトがない状態

オブジェクトが存在しないときに、[ボタン1]ボタンをクリックすると、「text」プロパティに「りんご」が入ったオブジェクトが生成される

オブジェクトが存在しないときに、[ボタン2]ボタンをクリックすると、「text」プロパティに「みかん」が入ったオブジェクトが生成される

JavaScript Sample - Windows Internet Explorer
ボタン1 ボタン2 内容の表示
結果:

りんご

みかん

りんご
りんご

みかん
みかん

[内容の表示]ボタンをクリックすると、現状のオブジェクトの「text」プロパティ値が表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
```

```
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <form action="./newobj.cgi" method="get" name="mainForm">
      <input type="button" id="makeButton1" value="ボタン1">
      <input type="button" id="makeButton2" value="ボタン2">
      <input type="button" id="makeButton3" value="内容の表示">
    </form>
    <div id="result">結果:</div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
var mainObj;
window.onload = function(){
  document.getElementById("makeButton1").onclick = function(){
    mainObj = mainObj || { "text" : "りんご" }; —1
    document.getElementById("result").innerHTML = mainObj.text;
  }
  document.getElementById("makeButton2").onclick = function(){
    mainObj = mainObj || { "text" : "みかん" }; —2
    document.getElementById("result").innerHTML = mainObj.text;
  }
  document.getElementById("makeButton3").onclick = function(){
    document.getElementById("result").innerHTML +=
    "<br>"+mainObj.text;                                          ]3
  }
}
```

1 …「mainObj」オブジェクトが存在しない場合には「{ "text" : "りんご" }」として新たに生成する

2 …「mainObj」オブジェクトが存在しない場合には「{ "text" : "みかん" }」として新たに生成する

3 …「mainObj」オブジェクトの「text」プロパティの内容を表示する

OnePoint オブジェクトの有無は「||」で判別する

オブジェクトが存在しない場合のみメソッド/プロパティを設定するには、「||」を使います。「||」の前に調べたいオブジェクト名を記述し、「||」の後ろにオブジェクトの定義を記述します。「||」は演算子で、「||」の左側が偽(false、null、undefined)であれば「||」以後の処理を行います。このため、すでにオブジェクトが定義されていれば「||」の前のオブジェクトが、そうでなければ以後に続く処理が行われることになります。

「||」で判別しなくても「if」を利用し、「if(!オブジェクト) オブジェクトの定義」とすることもできます。「||」を利用した場合にプログラムが読みにくいのであれば、「if」を使うのがよいでしょう。

SECTION

024 一定時間後に一度だけ処理を行う

ここでは、「setTimeout()」を使い、一定時間後に一度だけ処理を行う方法について解説します。

JavaScript Sample - Windows Internet Explorer

お勧めメニュー

ペペロンチーノ

プロシュート

モッツァレラチーズ

3秒後に段落内のスタイルが切り替わる

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>お勧めメニュー</h1>
    <p>ペペロンチーノ</p>
    <p>プロシュート</p>
```

```
    <p>モッツァレラチーズ</p>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  setTimeout("styleChange.paragraph()", 3*1000);—1
}
var styleChange = {
  paragraph : function(){
    var pTag = document.getElementsByTagName("p");
    for(var i=0; i<pTag.length; i++){
      pTag[i].className = "normal";
    }
  }
}
```

1 …「setTimeout()」で指定秒数後に実行される処理とミリ秒を指定する

OnePoint 指定秒数後に一回のみ処理を行うには「setTimeout()」を使う

「setTimeout()」を使えば、指定した処理を指定秒後に1回だけ呼び出すことができます。秒数はミリ秒で指定するため、1秒後に処理するのであれば「1000」を指定します。「3*1000」であれば3秒後に処理が実行されることになります。ただし、「setTimeout()」で指定した時間は厳密ではないため、動作時間に関しては保証されません。

「setTimeout()」で設定したタイマーを解除するには、「clearTimeout()」を使います。「setTimeout()」は、タイマーが設定されるとタイマーIDを返します。このタイマーIDを「clearTimeout()」のパラメータに指定することで、タイマーを停止させることができます。ただし、一度停止してしまったタイマーを再開させる機能はありません(一時停止の機能はない)。

なお、1回のみではなく定期的に処理を呼び出すには、「setInterval()」を使います(66ページ参照)。

Column Internet Explorerでの非同期通信でのデッドロックを回避する

Internet Explorerでまとめて非同期通信のリクエストを発行すると、一時的に画面の更新が停止してしまうことがあります。このような場合には、「setTimeout()」を利用して処理時間を空けることで回避することができます。なお、この現象は他のブラウザでは発生しません。

```
ajaxProccess()
```

↓

```
setTimeout("ajaxProccess()", 500);
```

025 一定時間ごとに処理を繰り返す

ここでは、「setInterval()」を使い、一定時間ごとに処理を繰り返し行う方法について解説します。

JavaScript Sample - Windows Internet Explorer

セール開催中

90%OFF

1秒ごとに段落の背景を変化させる

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; 
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css" 
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h3>セール開催中</h3>
    <p>90%OFF</p>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  setInterval("styleChange.bg()", 1*1000);
}
var styleChange = {
  bg : function(){
    var r = Math.floor(Math.random()*256);
    var g = Math.floor(Math.random()*256);
    var b = Math.floor(Math.random()*256);
    var rgb = "rgb("+r+","+g+","+b+")";
    document.getElementsByTagName("p")[0].style.backgroundColor = 
    rgb;
  }
}
```

1 …「setInterval()」で指定秒数ごとに実行する処理とミリ秒を指定する

2 …最初の段落の背景色を設定する

OnePoint 指定秒数ごとに処理を行うには「setInterval()」を使う

「setInterval()」を使えば、指定した処理を指定秒数ごと呼び出すことができます。秒数はミリ秒で指定するため、1秒ごとに処理するのであれば「1000」を指定します。「3*1000」であれば3秒ごとに処理が実行されることになります。ただし、「setInterval()」で指定した時間は厳密ではないため、動作時間に関しては保証されません。

「setInterval()」で設定したタイマーを解除するには、「clearInterval()」を使います。「setInterval()」はタイマーが設定されると、タイマーIDを返します。このタイマーIDを「clearInterval()」のパラメータに指定することで、タイマーを停止させることができます。ただし、一度停止してしまったタイマーを再開させる機能はありません(一時停止の機能はない)。

なお、1回だけ処理を呼び出すには、「setTimeout()」を使います(64ページ参照)。

Column JavaScript+スタイルシートでのカラー指定

従来、JavaScriptでページの背景色などのカラーを指定する場合には「カラー名」または「カラーコード」を使っていました。カラー名は「red」や「orange」などの色を示す英単語です。カラーコードは「#rrggbb」形式、つまり「#」の後に16進数で赤色、緑色、青色の明るさ(0〜255)を指定します。

JavaScriptでスタイルシートにアクセスして色指定を行う場合には、これに加えてスタイルシートでの色指定の形式を利用することができます。サンプルのように「rgb(赤の輝度, 緑の輝度, 青の輝度)」や「#rgb」形式などを指定することができます。

Break Time

複数のJavaScriptファイルを読み込む

JavaScriptのプログラムコードが書かれたファイルが複数ある場合には、<script>タグを列記して読み込ませるか、後付けで読み込ませる方法があります。

その他にも複数のスクリプトファイルを読み込ませるには、少し異なる手法もあります。次のサンプルのように「src」属性に読み込むメインファイル名の後ろに「?」を記述し、以後に「,」(カンマ)で区切って読み込むファイルを列記します。メインプログラム(main.js)では、次のサンプルのように指定します。

これは最初に書かれた<script>タグの「src」属性の文字列を読み込み、指定されたスクリプトファイルを読み込ませます。この手法を利用すればHTML文書内には1行だけ<script>タグを記述すればよいことになり、メンテナンス性も向上します。これはスタイルシートの「@import」で別ファイルを読み込むのと似ています。

●HTMLファイル

```
<script type="text/javascript" src="main.js?sub1.js,sub2.js">
</script>
```

●JavaScriptファイル(main.js)

```
(function (){
  var srcURL = document.getElementsByTagName("script")[0].src;
    var loadURL = srcURL.split("?")[1].split(",");
  for (var i=0; i<loadURL.length; i++) {
    document.write("<script src='"+loadURL[i]+"'></script>");
  }
})();
```

上記のサンプルを実行すると、sub1.jsとsub2.jsが実行されます。本書のサンプルの場合は、次のようにアラートが表示されます。

Windows Internet Explorer
sub1.jsです
OK

Windows Internet Explorer
sub2.jsです
OK

CHAPTER - 03

フォーム

SECTION 026 フォームのエレメントを制御する方法について

入力フォームの制御方法

入力フォームの制御(参照や設定)の方法はいくつか用意されていますが、安全性·確実性を求めるのであれば従来通りの書き方を踏襲するのがよいでしょう。

DOMを利用する方法でもフォームの制御は可能ですが、ブラウザによっては期待通りに動作しない場合があるため、フォーム名、エレメント名の「id」属性、「name」属性の指定には注意が必要です。どちらも同じ名前にしておくのもよいでしょう。また、Internet ExplorerではDOMでの制御を行おうとした場合に、正しく指定できずにエラーとなってしまうエレメントもあるため注意が必要です。

▶ フォームの基本的な制御方法

入力フォームの制御には、「forms」配列、エレメントの制御は「elements」配列を使います。「forms」「elements」は連想配列でもあるため、配列に参照番号または「name」属性で指定した名前をキーとして指定することもできます。配列の参照番号は、HTML文書内での出現順に番号が割り振られます。

▶ ラジオボタンの制御

ラジオボタンは、「name」属性で同じ名前が指定されます。JavaScriptではこれらがラジオボタングループとして扱われ、「name」属性をキーとしたラジオボタン名の配列として、各ラジオボタンを制御することができます。

▶ セレクトメニューの制御

セレクトメニューの場合、「options」配列を使用することで各項目を制御することができます。

JavaScript Sample - Windows Internet Explorer
ユーザー登録
氏名: 氏名を入力
住所: 住所を入力
メールアドレス: メールアドレスを入力
☑メールの受け取りを希望する
○男
◉女
このページをどこで知りましたか?
その他
このサービスは主にどこで利用しますか?
会社
自宅
その他

ページが読み込まれたときに、各エレメント、メニューに値を設定するサンプル

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
```

```
  <body>
    <h1>ユーザー登録</h1>
    <form action="./check.cgi" method="get" id="mainForm"
    name="mainForm">
      氏名:<input type="text" name="tfld1"><br>
      住所:<input type="text" name="tfld2"><br>
      メールアドレス:<input type="text" name="tfld3" id="tfld3"><br>
      <input type="checkbox" name="cb">メールの受け取りを希望する<br>
      <input type="radio" name="rd">男<br>
      <input type="radio" name="rd">女<br>
      このページをどこで知りましたか?<br>
      <select id="sel1" name="sel1">
        <option value="item1">Web</option>
        <option value="item2" name="second">その他</option>
      </select><br>
      このサービスは主にどこで利用しますか?<br>
      <select id="sel2" name="sel2" multiple>
        <option value="item1">会社</option>
        <option value="item2">自宅</option>
        <option value="item3">その他</option>
      </select>
    </form>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.forms[0].elements[0].value = "氏名を入力"; ―1
  document.forms["mainForm"].elements["tfld2"].value = "住所を入力"; ―2
  document.getElementById("tfld3").value = "メールアドレスを入力"; ―3
  document.mainForm.cb.checked = true; ―4
  document.mainForm.rd[1].checked = true; ―5
  document.forms["mainForm"]["sel1"].options[1].selected = true; ―6
  document.forms["mainForm"]["sel2"].options[1].selected = true; ―7
}
```

1 …「forms」配列と「elements」配列を使って制御する基本パターン

2 …「forms」「elements」は連想配列(ハッシュ)でも制御することができる

3 …特定のID名を持つエレメントを直接指定する場合は「document.getElementById()」を使用する

4 …「name」属性で指定した名前を「.」(ドット)で区切って指定することもできる

5 …ラジオボタンの場合は「name」属性で指定したグループ配列として制御する

6 …セレクトメニューの場合は「options」配列を使って各項目を制御する

7 …セレクトメニューの場合、複数選択できるメニューの場合でも、同様に「options」配列を使って制御する

SECTION 027 ページが読み込まれたら指定したテキストフィールドにフォーカスを移す

ここでは、ページの読み込み時に、指定したテキストフィールドにフォーカスを移す方法について解説します。

ページが表示されると、指定したテキストフィールドにフォーカスが移動する

SOURCE CODE　HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>IDとパスワードを入力してログインしてください。</h1>
    <form action="./check.cgi" method="get" name="mainForm">
      ID:<input type="text" name="userID" id="userID"><br> ―1
      PW:<input type="text" name="userPW" id="userPW"><br>
      <input type="button" id="loginButton" value="ログイン">
    </form>
  </body>
</html>
```

1 …フォーカスしたいエレメントにID名を指定する

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("userID").focus(); —1
}
```

1 …「focus()」を使って指定したエレメントにフォーカスを移す

OnePoint フォーカスするには「focus()」を使う

ページ上の特定のエレメントにフォーカスを移すには、「focus()」を使います。ページが読み込まれたら特定のテキストフィールドにフォーカスするには、「window.onload」イベントが発生した際にフォーカス処理を行うようにします。サンプルではID名がuserIDのテキストフィールドにフォーカスするようにしています。

Column フォーカスを外す

フォームのテキストフィールドにフォーカスされた状態からフォーカスを外すには、「blur()」を使います。サンプルの場合は、次のように「focus()」の代わりに「blur()」を指定します。

```
document.getElementById("userID").blur();
```

Column テキストフィールドの内容を選択する

ページが読み込まれたらテキストフィールドの内容を選択するには、「focus()」と「select()」を組み合わせます。次のように最初にフォーカスし、その後に選択するようにします。

```
window.onload = function(){
  document.getElementById("userID").focus();
  document.getElementById("userID").select();
}
```


ページが表示されると、指定したテキストフィールドの内容が選択される

028 テキストフィールドの内容をチェックする

ここでは、テキストフィールドの内容を調べ、特定の文字列と内容が一致するかをチェックする方法について解説します。

●IDが間違っている場合

JavaScript Sample - Windows Internet Explorer

IDの確認

ID: tokyo

IDをチェック

Windows Internet Explorer

IDが間違っています

OK

●IDが正しい場合

JavaScript Sample - Windows Internet Explorer

IDの確認

ID: cr

IDをチェック

Windows Internet Explorer

このIDでログインします

OK

IDを入力してボタンをクリックすると、入力されたIDが正しいかどうか判断する

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
```

```
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>IDの確認</h1>
    <form action="./check.cgi" method="get" name="mainForm">
      ID:<input type="text" name="userID" id="userID"><br> —1
      <input type="button" id="checkButton" value="IDをチェック">
    </form>
  </body>
</html>
```

1 …内容をチェックしたいエレメントにID名を指定する

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("checkButton").onclick = function(){
    var uID = document.getElementById("userID").value; —1
    if (uID != "cr"){
      alert("IDが間違っています");
    }else{                                           —2
      alert("このIDでログインします");
    }
  }
}
```

1 …ID名が「userID」のテキストフィールドの内容を読み出す

2 …読み出した内容が「cr」の文字かどうか調べメッセージを表示する

OnePoint 「value」プロパティでテキストフィールドやエレメントの内容を読み出しチェックする

テキストフィールドに入力された内容をチェックするには、「value」プロパティで内容を読み出し、「if」または「switch」などで判断を行います。サンプルでは入力された文字が「cr」かどうか調べて処理を分けています。

単純に空欄かどうか調べる場合には、次のように否定演算子「!」を使って判断します。

```
if (!uID){
  alert("空欄です");
}else{
  alert("文字が入力されています");
}
```

空欄が複数ある場合も同様ですが、prototype.jsライブラリを使用すると手軽に空欄チェックを行うことができます(46ページ参照)。

SECTION 029

テキストフィールドに数字以外が入力されているか調べる

ここでは、テキストフィールドの内容を調べ、その内容に数字以外が入力されているかチェックする方法について解説します。

JavaScript Sample - Windows Internet Explorer

出荷状況の確認

伝票番号: s0123 (半角の数字のみ入力可)

チェック

Windows Internet Explorer

半角の数字だけ入力してください

OK

入力された文字が0～9の数字でない場合にボタンをクリックすると、アラートダイアログが表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; ⏎
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css" ⏎
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>出荷状況の確認</h1>
    <form action="./check.cgi" method="get" name="mainForm">
      伝票番号:<input type="text" name="qNumber" ⏎
      id="qNumber">(半角の数字のみ入力可)<br>       ─ 1
      <input type="button" id="checkButton" value="チェック">
    </form>
  </body>
</html>
```

1 …内容をチェックしたいエレメントにID名を指定する

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("checkButton").onclick = function(){
    var text = document.getElementById("qNumber").value; ―1
    var result = text.match(/[^0-9]/); ―2
    if (!text || result) {
      alert("半角の数字だけ入力してください");
      document.getElementById("qNumber").focus();        ―3
      document.getElementById("qNumber").select();
    }
  }
}
```

1 …ID名が「qNumber」のテキストフィールドの内容を読み出す

2 …読み出した内容を正規表現を使って数値以外が含まれるかどうか調べる

3 …テキストフィールドが空欄または数字以外が入力されている場合にアラートダイアログを表示し、フォーカスをテキストフィールドに移す

OnePoint 数値かどうかのチェックは正規表現を使う

テキストフィールドに入力されたデータが数値かどうかを調べるには、正規表現を使います。JavaScriptでは「match()」で正規表現の文字列を指定することができ、結果が配列で返されます。もし、何もマッチしなかった場合には、「match()」は「null」を返します。

数値かどうか調べる場合にはマッチする条件を数値にマッチではなく、数値以外にマッチするように指定します。数値以外にマッチさせるための正規表現指定は「match(/[^0-9]/)となります。「^」は、「～以外」という意味になります。「[^0-9]」とすると、0から9以外にマッチすることになります。

Column ページ内容を印刷する

JavaScriptでは「print()」メソッドが用意されており、任意のウィンドウを印刷することができます。次のサンプルはID名「printButton」のボタンがクリックされたら現在のウィンドウの内容を印刷するプログラムです。

```
window.onload = function(){
  document.getElementById("printButton").onclick = function(){
    window.print();
  }
}
```

印刷前には「onbeforeprint」イベントと「onafterprint」イベントが発生します。印刷データ生成前に「onbeforeprint」が発生するので、表示したくないエレメントやテキストを非表示にし、印刷データ生成後に発生する「onafterprint」イベントで元の状態に復帰させるような処理を行うことができます。

SECTION

030 テキストフィールドに英文字以外が入力されているか調べる

ここでは、テキストフィールドの内容を調べ、その内容に英文字以外が入力されているかチェックする方法について解説します。

入力された文字が英文字でない場合にボタンをクリックすると、アラートダイアログを表示する

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>キーの確認</h1>
    <form action="./check.cgi" method="get" name="mainForm">
      キー:<input type="text" name="qCode" id="qCode"><br> ―1
      (半角の英字のみ入力可)<br>
      <input type="button" id="checkButton" value="チェック">
    </form>
  </body>
</html>
```

1 …内容をチェックしたいエレメントにID名を指定する

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("checkButton").onclick = function(){
    var text = document.getElementById("qCode").value; ―1
    var result = text.match(/[^a-z]/i); ―2
    if (!text || result) {
      alert("a～zの英文字だけ入力してください");
      document.getElementById("qCode").focus();          3
      document.getElementById("qCode").select();
    }
  }
}
```

1 …ID名が「qCode」のテキストフィールドの内容を読み出す

2 …読み出した内容を正規表現を使って英大文字、英小文字以外が含まれるかどうか調べる

3 …テキストフィールドが空欄または英字以外が入力されている場合にアラートダイアログを表示し、フォーカスをテキストフィールドに移す

OnePoint 英文字かどうかのチェックは正規表現を使う

テキストフィールドに入力されたデータが英文字かどうかを調べるには、正規表現を使います。JavaScriptでは「match()」で正規表現の文字列を指定することができ、結果が配列で返されます。もし、何もマッチしなかった場合には、「match()」は「null」を返します。

英文字かどうか調べる場合にはマッチする条件を英文字にマッチではなく、英文字以外にマッチするように指定します。英文字以外にマッチさせるための正規表現指定は、「match(/[^a-z]/)となります。「^」は「～以外」という意味になります。「[^a-z]」とすると、英文字のaからz以外にマッチすることになります。

また、英文字の場合、小文字と大文字がありこれらを区別せずにチェックしたい場合には、「i」オプション(ignore)を指定します。これは、「match(/[^a-z]/i)」のように記述します。

Column 無限ループや一定時間内に処理が終わらない場合

作成したプログラムによっては条件判断の間違いで処理が終わらなくなってしまったり、アルゴリズムが悪くて処理に時間がかかってしまう場合があります。古いブラウザでは強制終了するといった方法がとられていましたが、現在利用されている多くのブラウザでは一定時間内に処理が終わらないと警告が表示されるようになっています。

◉警告画面

SECTION 031 テキストフィールドの内容が間違っている場合一瞬だけハイライトさせる

ここでは、テキストフィールドの内容を調べ、その内容が特定の文字列と一致しない場合に、script.aculo.usライブラリを使って一瞬だけハイライトさせる方法について解説します。

JavaScript Sample - Windows Internet Explorer

IDの確認

ID: osaka

IDをチェック

テキストフィールドにID名を入力し、ボタンをクリックすると…

IDが正しくない場合にテキストフィールドがハイライト表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type"
    content="text/html;charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
1   <script type="text/javascript" src="prototype.js"></script>
    <script type="text/javascript" src="scriptaculous.js"></script>
```

```
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>IDの確認</h1>
    <form action="./check.cgi" method="get" name="mainForm">
      ID:<input type="text" name="userID" id="userID"><br> —2
      <input type="button" id="checkButton" value="IDをチェック">
    </form>
  </body>
</html>
```

1 …prototype.jsライブラリとscript.aculo.usライブラリを読み込む
2 …内容をチェックしたいエレメントにID名を指定する

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("checkButton").onclick = function(){
    var uID = document.getElementById("userID").value; —1
    if (uID != "cr"){ —2
      new Effect.Highlight("userID"); —3
    }else{
      alert("正しいIDが入力されました");
    }
  }
}
```

1 …ID名が「userID」のテキストフィールドの内容を読み出す
2 …入力されたIDの文字を調べる
3 …正しくない場合にはテキストフィールドをハイライトする

OnePoint 「Effect.Highlight()」でハイライトさせる

テキストフィールドに入力されたデータが不正な場合や空欄の場合に、テキストフィールドをハイライト表示させるには、script.aculo.usライブラリを使うと便利です。script.aculo.usライブラリはprototype.jsライブラリに依存しているので、両方のライブラリを読み込ませます。

ハイライト表示するには、「Effect.Highlight()」のパラメータにハイライト表示させたいテキストフィールドのID名またはオブジェクトを指定するだけです。後は、自動的にライブラリが処理を行ってくれます。

032 パスワードが間違っている場合に左右に揺らす

ここでは、テキストフィールドの内容を調べ、その内容が特定の文字列と一致しない場合に、script.aculo.usライブラリを使ってウィンドウを左右に揺らす方法について解説します。

テキストフィールドにパスワードを入力してボタンをクリックすると…

パスワードが正しくない場合に入力フォーム全体が左右に揺れる

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type"
    content="text/html;charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="prototype.js"></script>
    <script type="text/javascript" src="scriptaculous.js"></script>
```

```
    <script type="text/javascript" src="main.js"></script>
  </head>

  <body>
    <h1>パスワードの確認</h1>
    <form action="./check.cgi" method="get" id="mainForm"
    name="mainForm">
      PW:<input type="password" name="pwd" id="pwd"><br> ―2
      <input type="button" id="checkButton" value="PWをチェック">
    </form>
  </body>
</html>
```

1 …prototype.jsライブラリとscript.aculo.usライブラリを読み込む

2 …内容をチェックしたいエレメントにID名を指定する

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("checkButton").onclick = function(){
    var pw = document.getElementById("pwd").value; ―1
    if (pw != "cr"){ ―2
      new Effect.Shake("mainForm"); ―3
    }else{
      alert("正しいパスワードが入力されました");
    }
  }
}
```

1 …ID名が「pwd」のテキストフィールドの内容を読み出す

2 …入力された文字列(パスワード)を調べる

3 …正しくない場合には入力フォーム全体を左右に揺らす

OnePoint 「Effect.Shake()」で入力エリアを揺らす

パスワードが間違って入力された場合に入力フォームを左右に揺らして間違いだと知らせるには、script.aculo.usライブラリを使うと便利です。script.aculo.usライブラリはprototype.jsライブラリに依存しているので、両方のライブラリを読み込ませます。

入力フォームを左右に揺らすには、「Effect.Shake()」のパラメータに揺らしたい入力フォームのID名またはオブジェクトを指定するだけです。後は、自動的にライブラリが処理を行ってくれます。

033 数値以外が入力されたらメッセージを表示する

ここでは、Adobe Spryフレームワークを使って、数値以外が入力されたらメッセージを表示する方法について解説します。

テキストフィールドに数値以外の文字が入力されると、メッセージが表示される

SOURCE CODE　HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" href="SpryValidationTextField.css"
    type="text/css" media="all">
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
```

```
    <script src="SpryValidationTextField.js"
    type="text/javascript"></script>
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <form action="./check.cgi" method="get" id="mainForm"
    name="mainForm">
      <span id="checkData1">
        <input type="text" name="text1" id="text1">
        <span class="textfieldRequiredMsg">数値を入力してください</span>
        <span class="textfieldInvalidFormatMsg">
        数値以外は受け付けません</span>
      </span>
      <br>
      <input type="submit" value="送信">
    </form>
  </body>
</html>
```

1 …Adobe Spryフレームワークで使用するスタイルシートファイルを読み込む

2 …Adobe Spryフレームワークを読み込む

SOURCE CODE JavaScript main.js

```
window.onload = function() {
  new Spry.Widget.ValidationTextField("checkData1", "real",
  {validateOn:["blur","change"]});
}
```

1 …ID名が「checkData1」で囲まれた範囲にあるテキストフィールドの内容をチェックするように設定する

OnePoint 数値かどうかチェックを行うには「Spry.Widget.ValidationTextField()」で「real」を指定する

リアルタイムに入力された内容が数値かどうかを調べるには、Adobe Spryフレームワークを利用すると簡単です。テキストフィールドのチェックを行うSpryValidationTextField.jsファイルと関連するスタイルシートファイル(SpryValidationTextField.css)を読み込みます。チェックしたいテキストフィールドを<span>タグで囲みID名を指定しておきます。さらに入力時に表示するメッセージを<span>タグで囲み、スタイルシートのクラス名を「textfieldRequiredMsg」にします。入力内容が不正だった場合のメッセージも同様に<span>タグで囲み、スタイルシートクラス名を「textfieldInvalidFormatMsg」にします。

HTML文書の設定が終わったらスクリプトでチェックする対象となるテキストフィールドを「Spry.Widget.ValidationTextField()」で指定します。パラメータにはチェックする領域を示すID名、チェックする種類、いつチェックするかなどのオプションを指定します。数値かどうかをチェックする場合には2番目のパラメータに「real」の文字を指定します。これでリアルタイムに数値かどうかの入力チェックが行われます。なお、「real」の代わりに「integer」を指定すると、整数かどうかの入力チェックが行われます。

金額以外が入力されたらメッセージを表示する

ここでは、Adobe Spryフレームワークを使って、金額以外が入力されたらメッセージを表示する方法について解説します。

JavaScript Sample - Windows Internet Explorer

落札希望金額

金額: 金額を入力してください
送信

コンピュータ | 保護モード: 無効

テキストフィールドに金額以外の文字が入力されると、メッセージが表示される

JavaScript Sample - Windows Internet Explorer

落札希望金額

金額: 1,23 金額以外は受け付けません
送信

コンピュータ | 保護モード: 無効 100%

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" href="SpryValidationTextField.css"
    type="text/css" media="all">                                  1
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
```

```
    <script src="SpryValidationTextField.js"
    type="text/javascript"></script>
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>落札希望金額</h1>
    <form action="./check.cgi" method="get" id="mainForm"
    name="mainForm">
      <span id="checkData1">
        金額:<input type="text" name="text1" id="text1">
        <span class="textfieldRequiredMsg">金額を入力してください</span>
        <span class="textfieldInvalidFormatMsg">
        金額以外は受け付けません</span>
      </span>
      <br>
      <input type="submit" value="送信">
    </form>
  </body>
</html>
```

1 …Adobe Spryフレームワークで使用するスタイルシートファイルを読み込む

2 …Adobe Spryフレームワークを読み込む

SOURCE CODE JavaScript main.js

```
window.onload = function() {
  new Spry.Widget.ValidationTextField("checkData1", "currency",
  {validateOn:["blur","change"]});
}
```

1 …ID名が「checkData1」で囲まれた範囲にあるテキストフィールドの内容をチェックするように設定する

OnePoint 金額かどうかチェックを行うには「Spry.Widget.ValidationTextField()」で「currency」を指定する

リアルタイムに入力された内容が金額かどうかを調べるには、Adobe Spryフレームワークを利用すると簡単です。テキストフィールドのチェックを行うSpryValidationTextField.jsファイルと関連するスタイルシートファイル(SpryValidationTextField.css)を読み込みます。チェックしたいテキストフィールドを<span>タグで囲み、ID名を指定しておきます。さらに入力時に表示するメッセージを<span>タグで囲み、スタイルシートのクラス名を「textfieldRequiredMsg」にします。入力内容が不正だった場合のメッセージも同様に<span>タグで囲み、スタイルシートクラス名を「textfieldInvalidFormatMsg」にします。

HTML文書の設定が終わったらスクリプトでチェックする対象となるテキストフィールドを「Spry.Widget.ValidationTextField()」で指定します。パラメータにはチェックする領域を示すID名、チェックする種類、いつチェックするかなどのオプションを指定します。金額かどうかをチェックする場合には2番目のパラメータに「currency」の文字を指定します。これでリアルタイムに金額かどうかの入力チェックが行われます。

SECTION

035 日付以外が入力されたらメッセージを表示する

ここでは、Adobe Spryフレームワークを使って、日付以外が入力されたらメッセージを表示する方法について解説します。

テキストフィールドに日付以外の文字が入力されると、メッセージが表示される

SOURCE CODE　HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" href="SpryValidationTextField.css"
    type="text/css" media="all">                              1
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script src="SpryValidationTextField.js"
    type="text/javascript"></script>                           2
    <script type="text/javascript" src="main.js"></script>
  </head>
```

```
  <body>
    <h1>生年月日の確認</h1>
    <form action="./check.cgi" method="get" id="mainForm" 
    name="mainForm">
      <span id="checkData1">
        生年月日：<input type="text" name="text1" id="text1">
        <span class="textfieldRequiredMsg">
        yyyy/mm/dd形式で入力してください</span>
        <span class="textfieldInvalidFormatMsg">
        yyyy/mm/dd形式以外は受け付けません</span>
      </span>
      <br>
      <input type="submit" value="送信">
    </form>
  </body>
</html>
```

1 …Adobe Spryフレームワークで使用するスタイルシートファイルを読み込む

2 …Adobe Spryフレームワークを読み込む

SOURCE CODE　JavaScript main.js

```
window.onload = function() {
  new Spry.Widget.ValidationTextField("checkData1", "date", 
     {format:"yyyy/mm/dd", validateOn:["blur","change"]});   1
}
```

1 …ID名が「checkData1」で囲まれた範囲にあるテキストフィールドの内容をチェックするように設定する

OnePoint 日付かどうかチェックするには「Spry.Widget.ValidationTextField()」で「date」とフォーマットを指定する

リアルタイムに入力された内容が指定された形式の日付かどうかを調べるには、Adobe Spryフレームワークを利用すると簡単です。テキストフィールドのチェックを行うSpryValidationTextField.jsファイルと関連するスタイルシートファイル（SpryValidationTextField.css）を読み込みます。チェックしたいテキストフィールドを<span>タグで囲み、ID名を指定しておきます。さらに入力時に表示するメッセージを<span>タグで囲み、スタイルシートのクラス名を「textfieldRequiredMsg」にします。入力内容が不正だった場合のメッセージも同様に<span>タグで囲み、スタイルシートクラス名を「textfieldInvalidFormatMsg」にします。

HTML文書の設定が終わったらスクリプトでチェックする対象となるテキストフィールドを「Spry.Widget.ValidationTextField()」で指定します。パラメータにはチェックする領域を示すID名、チェックする種類、いつチェックするかなどのオプションを指定します。日付かどうかをチェックする場合には、2番目のパラメータに「date」の文字を指定します。次にどのような日付形式かどうかを3番目の「format」オプションで指定します。年は「yy」、月は「mm」、日にちは「dd」で示されます。「mm/dd/yy」であれば「10/19/07」（2007年10月19日）、「yyyy/mm/dd」であれば「2007/10/19」の形式を受け付けるようになります。

SECTION 036

時刻以外が入力されたらメッセージを表示する

ここでは、Adobe Spryフレームワークを使って、時刻以外が入力されたらメッセージを表示する方法について解説します。

テキストフィールドに時刻以外の文字が入力されると、メッセージが表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; ↩
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" href="SpryValidationTextField.css" ↩
    type="text/css" media="all">                              1
    <link rel="stylesheet" type="text/css" href="main.css" ↩
    media="all">
    <script src="SpryValidationTextField.js" ↩
    type="text/javascript"></script>                          2
    <script type="text/javascript" src="main.js"></script>
  </head>
```

```
  <body>
    <h1>今朝の起床時刻</h1>
    <form action="./check.cgi" method="get" id="mainForm" 
    name="mainForm">
      <span id="checkData1">
        時刻:<input type="text" name="text1" id="text1">
        <span class="textfieldRequiredMsg">
        HH:MM:SS形式で入力してください</span>
        <span class="textfieldInvalidFormatMsg">
        HH:MM:SS形式以外は受け付けません</span>
      </span>
      <br>
      <input type="submit" value="送信">
    </form>
  </body>
</html>
```

1 …Adobe Spryフレームワークで使用するスタイルシートファイルを読み込む
2 …Adobe Spryフレームワークを読み込む

SOURCE CODE　JavaScript main.js

```
window.onload = function() {
  new Spry.Widget.ValidationTextField("checkData1", "time", 
    {format:"hh:mm:ss", validateOn:["blur","change"]});   1
}
```

1 …ID名が「checkData1」で囲まれた範囲にあるテキストフィールドの内容をチェックするように設定する

OnePoint 時刻かどうかをチェックするには「Spry.Widget.ValidationTextField()」で「time」とフォーマットを指定する

リアルタイムに入力された内容が指定された形式の時刻かどうかを調べるには、Adobe Spryフレームワークを利用すると簡単です。テキストフィールドのチェックを行うSpryValidationTextField.jsファイルと関連するスタイルシートファイル(SpryValidationTextField.css)を読み込みます。チェックしたいテキストフィールドを<span>タグで囲み、ID名を指定しておきます。さらに入力時に表示するメッセージを<span>タグで囲み、スタイルシートのクラス名を「textfieldRequiredMsg」にします。入力内容が不正だった場合のメッセージも同様に<span>タグで囲み、スタイルシートクラス名を「textfieldInvalidFormatMsg」にします。

HTML文書の設定が終わったら、スクリプトでチェックする対象となるテキストフィールドを「Spry.Widget.ValidationTextField()」で指定します。パラメータにはチェックする領域を示すID名、チェックする種類、いつチェックするかなどのオプションを指定します。時刻かどうかをチェックする場合には、2番目のパラメータに「time」の文字を指定します。次にどのような時刻形式かどうかを3番目の「format」オプションで指定します。時は「hh」、分は「mm」、秒は「ss」で示されます。「hh:mm:ss」であれば、「11:16:35」(11時16分35秒)の形式を受け付けることになります。また、タイムゾーンも許可する場合には、「tt」を使って指定します。

SECTION

037 IPアドレス以外が入力されたらメッセージを表示する

ここでは、Adobe Spryフレームワークを使って、IPアドレス以外が入力されたらメッセージを表示する方法について解説します。

JavaScript Sample - Windows Internet Explorer

IPアドレスの確認

IPアドレス: [ ] IPアドレスを入力してください
送信

テキストフィールドにIPアドレス以外の文字が入力されると、メッセージが表示される

JavaScript Sample - Windows Internet Explorer

IPアドレスの確認

IPアドレス: 1923 IPアドレス以外は受け付けません
送信

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; ⤶
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" href="SpryValidationTextField.css" ⤶   ]1
    type="text/css" media="all">
    <link rel="stylesheet" type="text/css" href="main.css" ⤶
    media="all">
    <script src="SpryValidationTextField.js" ⤶   ]2
    type="text/javascript"></script>
    <script type="text/javascript" src="main.js"></script>
  </head>
```

```
  <body>
    <h1>IPアドレスの確認</h1>
    <form action="./check.cgi" method="get" id="mainForm"
    name="mainForm">
      <span id="checkData1">
        IPアドレス:<input type="text" name="text1" id="text1">
        <span class="textfieldRequiredMsg">
        IPアドレスを入力してください</span>
        <span class="textfieldInvalidFormatMsg">
        IPアドレス以外は受け付けません</span>
      </span>
      <br>
      <input type="submit" value="送信">
    </form>
  </body>
</html>
```

1 …Adobe Spryフレームワークで使用するスタイルシートファイルを読み込む

2 …Adobe Spryフレームワークを読み込む

SOURCE CODE JavaScript main.js

```
window.onload = function() {
  new Spry.Widget.ValidationTextField("checkData1", "ip",
  {validateOn:["blur","change"]});    1
}
```

1 …ID名が「checkData1」で囲まれた範囲にあるテキストフィールドの内容をチェックするように設定する

OnePoint IPアドレスかどうかチェックするには「Spry.Widget.ValidationTextField()」で「ip」を指定する

リアルタイムに入力された内容がIPアドレスかどうかを調べるには、Adobe Spryフレームワークを利用すると簡単です。テキストフィールドのチェックを行うSpryValidationTextField.jsファイルと関連するスタイルシートファイル(SpryValidationTextField.css)を読み込みます。チェックしたいテキストフィールドを<span>タグで囲み、ID名を指定しておきます。さらに入力時に表示するメッセージを<span>タグで囲み、スタイルシートのクラス名を「textfieldRequiredMsg」にします。入力内容が不正だった場合のメッセージも同様に<span>タグで囲み、スタイルシートクラス名を「textfieldInvalidFormatMsg」にします。

HTML文書の設定が終わったらスクリプトでチェックする対象となるテキストフィールドを「Spry.Widget.ValidationTextField()」で指定します。パラメータにはチェックする領域を示すID名、チェックする種類、いつチェックするかなどのオプションを指定します。IPアドレスかどうかをチェックする場合には2番目のパラメータに「ip」の文字を指定します。これでリアルタイムにIPアドレスかどうかの入力チェックが行われます。

SECTION

038 URL以外が入力されたらメッセージを表示する

ここでは、Adobe Spryフレームワークを使って、URL以外が入力されたらメッセージを表示する方法について解説します。

JavaScript Sample - Windows Internet Explorer

お勧めサイト登録

URL: URLを入力してください

登録

テキストフィールドにURL以外の文字が入力されると、メッセージが表示される

JavaScript Sample - Windows Internet Explorer

お勧めサイト登録

URL: www.c- URL以外は受け付けません

登録

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; 
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" href="SpryValidationTextField.css"   ] 1
    type="text/css" media="all">
    <link rel="stylesheet" type="text/css" href="main.css" 
    media="all">
    <script src="SpryValidationTextField.js"   ] 2
    type="text/javascript"></script>
    <script type="text/javascript" src="main.js"></script>
  </head>
```

```
    <body>
      <h1>お勧めサイト登録</h1>
      <form action="./check.cgi" method="get" id="mainForm"
      name="mainForm">
        <span id="checkData1">
          URL:<input type="text" name="text1" id="text1">
          <span class="textfieldRequiredMsg">
          URLを入力してください</span>
          <span class="textfieldInvalidFormatMsg">
          URL以外は受け付けません</span>
        </span>
        <br>
        <input type="submit" value="登録">
      </form>
    </body>
</html>
```

1 …Adobe Spryフレームワークで使用するスタイルシートファイルを読み込む
2 …Adobe Spryフレームワークを読み込む

SOURCE CODE JavaScript main.js

```
window.onload = function() {
  new Spry.Widget.ValidationTextField("checkData1", "url",
  {validateOn:["blur","change"]});
}
```

1 …ID名が「checkData1」で囲まれた範囲にあるテキストフィールドの内容をチェックするように設定する

OnePoint URLかどうかをチェックするには「Spry.Widget.ValidationTextField()」で「url」を指定する

リアルタイムに入力された内容がURLかどうかを調べるには、Adobe Spryフレームワークを利用すると簡単です。テキストフィールドのチェックを行うSpryValidationTextField.jsファイルと関連するスタイルシートファイル(SpryValidationTextField.css)を読み込みます。チェックしたいテキストフィールドを<span>タグで囲み、ID名を指定しておきます。さらに入力時に表示するメッセージを<span>タグで囲み、スタイルシートのクラス名を「textfieldRequiredMsg」にします。入力内容が不正だった場合のメッセージも同様に<span>タグで囲み、スタイルシートクラス名を「textfieldInvalidFormatMsg」にします。

HTML文書の設定が終わったらスクリプトでチェックする対象となるテキストフィールドを「Spry.Widget.ValidationTextField()」で指定します。パラメータにはチェックする領域を示すID名、チェックする種類、いつチェックするかなどのオプションを指定します。URLかどうかをチェックする場合には2番目のパラメータに「url」の文字を指定します。これでリアルタイムにURLかどうかの入力チェックが行われます。Adobe Spryフレームワークで許されているのは、「http://」で始まるアドレスか、「ftp://」で始まるアドレスになります。

SECTION 039 メールアドレス以外が入力されたらメッセージを表示する

ここでは、Adobe Spryフレームワークを使って、メールアドレス以外が入力されたらメッセージを表示する方法について解説します。

テキストフィールドにメールアドレス以外の文字が入力されると、メッセージが表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" href="SpryValidationTextField.css"
    type="text/css" media="all">                                    1
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script src="SpryValidationTextField.js"
    type="text/javascript"></script>                               2
    <script type="text/javascript" src="main.js"></script>
  </head>
```

```
  <body>
    <h1>メールマガジンの読者登録</h1>
    <form action="./check.cgi" method="get" id="mainForm"
    name="mainForm">
      <span id="checkData1">
        メールアドレス:<input type="text" name="text1" id="text1">
        <span class="textfieldRequiredMsg">
        メールアドレスを入力してください</span>
        <span class="textfieldInvalidFormatMsg">
        メールアドレス以外は受け付けません</span>
      </span>
      <br>
      <input type="submit" value="登録">
    </form>
  </body>
</html>
```

1 …Adobe Spryフレームワークで使用するスタイルシートファイルを読み込む

2 …Adobe Spryフレームワークを読み込む

SOURCE CODE JavaScript main.js

```
window.onload = function() {
  new Spry.Widget.ValidationTextField("checkData1", "email",
  {validateOn:["blur","change"]});   ← 1
}
```

1 …ID名が「checkData1」で囲まれた範囲にあるテキストフィールドの内容をチェックするように設定する

OnePoint メールアドレスかどうかをチェックするには「Spry.Widget.ValidationTextField()」で「email」を指定する

リアルタイムに入力された内容がメールアドレスかどうかを調べるには、Adobe Spryフレームワークを利用すると簡単です。テキストフィールドのチェックを行うSpryValidationTextField.jsファイルと関連するスタイルシートファイル(SpryValidationTextField.css)を読み込みます。チェックしたいテキストフィールドを<span>タグで囲み、ID名を指定しておきます。さらに入力時に表示するメッセージを<span>タグで囲み、スタイルシートのクラス名を「textfieldRequiredMsg」にします。入力内容が不正だった場合のメッセージも同様に<span>タグで囲み、スタイルシートクラス名を「textfieldInvalidFormatMsg」にします。

HTML文書の設定が終わったらスクリプトでチェックする対象となるテキストフィールドを「Spry.Widget.ValidationTextField()」で指定します。パラメータにはチェックする領域を示すID名、チェックする種類、いつチェックするかなどのオプションを指定します。メールアドレスかどうかをチェックする場合には、2番目のパラメータに「email」の文字を指定します。これでリアルタイムにメールアドレスかどうかの入力チェックが行われます。

SECTION

040 入力パターンを指定する

ここでは、Adobe Spryフレームワークを使って、特定の形式だけを入力できるように制限する方法について解説します。

JavaScript Sample - Windows Internet Explorer

在庫確認

商品コード: ABC-000形式で入力してください

確認

テキストフィールドにABC-nnn-OOO以外のパターンを入力制限する（nは数値）

JavaScript Sample - Windows Internet Explorer

在庫確認

商品コード: ABC- ABC-000形式以外は受け付けません

確認

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" href="SpryValidationTextField.css"
    type="text/css" media="all">                                  1
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script src="SpryValidationTextField.js"
    type="text/javascript"></script>                              2
    <script type="text/javascript" src="main.js"></script>
  </head>
```

```
    <body>
      <h1>在庫確認</h1>
      <form action="./check.cgi" method="get" id="mainForm" ↩
      name="mainForm">
        <span id="checkData1">
          商品コード:<input type="text" name="text1" id="text1">
          <span class="textfieldRequiredMsg"> ↩
          ABC-000形式で入力してください</span>
          <span class="textfieldInvalidFormatMsg"> ↩
          ABC-000形式以外は受け付けません</span>
        </span>
        <br>
        <input type="submit" value="確認">
      </form>
    </body>
</html>
```

1 …Adobe Spryフレームワークで使用するスタイルシートファイルを読み込む

2 …Adobe Spryフレームワークを読み込む

SOURCE CODE JavaScript main.js

```
window.onload = function() {
  new Spry.Widget.ValidationTextField("checkData1", "custom",
    {pattern:"ABC-000-¥¥X¥¥Y¥¥Z",useCharacterMasking:true, ↩
    validateOn:["blur","change"]});                          1
}
```

1 …ID名が「checkData1」で囲まれた範囲にあるテキストフィールドの内容をチェックするように設定する

OnePoint 特定の形式で入力制限を行うには「Spry.Widget.ValidationTextField()」で「custom」と入力パターンを指定する

リアルタイムに入力された内容を制限するには、Adobe Spryフレームワークを利用すると簡単です。テキストフィールドのチェックを行うSpryValidationTextField.jsファイルと関連するスタイルシートファイル(SpryValidationTextField.css)を読み込みます。チェックしたいテキストフィールドを<span>タグで囲み、ID名を指定しておきます。さらに入力時に表示するメッセージを<span>タグで囲みスタイルシートのクラス名を「textfieldRequiredMsg」にします。入力内容が不正だった場合のメッセージも同様に<span>タグで囲み、スタイルシートクラス名を「textfieldInvalidFormatMsg」にします。

HTML文書の設定が終わったらスクリプトでチェックする対象となるテキストフィールドを「Spry.Widget.ValidationTextField()」で指定します。パラメータにはチェックする領域を示すID名、チェックする種類、いつチェックするかなどのオプションを指定します。特定の入力パターンのみ受け付けるようにする場合には、2番目のパラメータに「custom」の文字を指定します。次に3番目のオプションで入力パターンを指定します。これは、「pattern」を使って入力可能な文字列を指定します。強制的に文字を挿入/表示するには「¥¥」の後ろに表示する文字を1文字ずつ指定し、「useCharacterMasking」で「true」を指定します。

SECTION 041

必要な項目がすべて入力されていたら送信する

ここでは、必要な項目がすべて入力されているかどうかをチェックし、すべて入力されていたら、送信できるようにする方法について解説します。

*印が付いたテキストフィールドが未入力の場合にボタンをクリックすると…

アラートダイアログを表示して送信は行わない

SOURCE CODE　HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; 
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css" 
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>ユーザー登録</h1>
    <form action="./regist.cgi" method="get" id="mainForm" 
    name="mainForm">
```

```
      名　前:<input type="text" name="userName" id="userName">*<br>
      年　齢:<input type="text" name="userAge" id="userAge">*<br>
 1    住　所:<input type="text" name="userAddress" id="userAddress">*<br>
      メール:<input type="text" name="userEmail" id="userEmail"><br>
      ブログ:<input type="text" name="userBlog" id="userBlog"><br>
      *は必須項目<br>
      <input type="submit" id="loginButton" value="送信">
    </form>
  </body>
</html>
```

1 …入力チェックを行うテキストフィールドにID名を指定しておく

SOURCE CODE　JavaScript main.js

```
window.onload = function(){
  document.getElementById("mainForm").onsubmit = function(){
    var checkList = ["userName", "userAge", "userAddress"]; ―1
    for (var i=0; i<checkList.length; i++){                  ┐2
      if(!document.getElementById(checkList[i]).value){      ┘
        alert("未入力の項目があります");  ┐3
        return false;                      ┘
      }
    }
    return true; ―4
  }
}
```

1 …入力チェックするテキストフィールドのID名を配列に入れる

2 …配列の数だけ繰り返してテキストフィールドの内容を読み出し、空欄かどうか調べる

3 …空欄が1つでもあった場合はアラートダイアログを表示し、「return false」として送信しないようにする

4 …必須項目が入力されていた場合には「return true」としてデータの送信を行う

OnePoint チェックするテキストフィールドのID名を配列に入れておく

テキストフィールドの空欄チェックは、「value」プロパティを読み出し文字が入力されているかどうかを調べます。テキストフィールドが1つの場合は簡単ですが、複数ある場合にはあらかじめ調べるテキストフィールドのID名を配列に入れておきます。配列の数だけ繰り返して空欄チェックを行えば簡単なプログラムですみます。

SECTION 042 [同意します]のラジオボタンがONになったら送信ボタンを有効にする

ここでは、特定のラジオボタンがONになったときのみ、送信ボタンを有効にする方法について解説します。

JavaScript Sample - Windows Internet Explorer

使用許諾契約書の確認

本書に掲載されているサンプルを使用して発生した損害については責任を負わないものとします。

同意しません
同意します
次へ

[同意します]のラジオボタンがONになると、[次へ]ボタンが有効になる

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
```

```
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>使用許諾契約書の確認</h1>
    <form action="./confirm.cgi" method="get" id="mainForm" 
    name="mainForm">
      <textarea cols="30" rows="4">本書に掲載されているサンプルを使用して発生した
      損害については責任を負わないものとします。</textarea><br>
      <label><input type="radio" name="confirm" 
      id="NG">同意しません</label><br>
      <label><input type="radio" name="confirm" 
      id="Accept">同意します</label><br>
      <input type="submit" id="nextButton" value="次へ" disabled> ―1
    </form>
  </body>
</html>
```

1 …あらかじめボタンは無効にしておく

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("Accept").onclick = function(){       ┐
    document.getElementById("nextButton").disabled = false;     ┘1
  }
  document.getElementById("NG").onclick = function(){           ┐
    document.getElementById("nextButton").disabled = true;      ┘2
  }
}
```

1 …[同意します]のラジオボタンがクリックされたら[次へ]ボタンを有効にする
2 …[同意しません]のラジオボタンがクリックされたら[次へ]ボタンを無効にする

OnePoint ラジオボタンの選択状況に応じて「disabled」プロパティを操作する

　特定のラジオボタンが選択されたらボタンを有効にするには、「disabled」プロパティを操作します。サンプルでは[同意します]のラジオボタンがクリックされたら[次へボタン]の「disabled」プロパティに「false」を指定しクリックできるようにしています。逆に[同意しません]のラジオボタンがクリックされた場合には、「次へボタン」の「disabled」プロパティに「true」を指定しクリックできないようにします。

SECTION

043 どのラジオボタンがONになっているか調べる

ここでは、どのラジオボタンがONになっているか調べ、その項目名をページ上に表示する方法について解説します。

ラジオボタンの1つをONにして[チェック]ボタンをクリックすると…

ONになっているラジオボタンの項目名が表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
```

```html
  <body>
    <h1>職業をお答えください</h1>
    <form action="./enq.cgi" method="get" id="mainForm" 
    name="mainForm">
      <label><input type="radio" name="work" value="農家"> 
      農家</label><br>
      <label><input type="radio" name="work" value="会社員"> 
      会社員</label><br>
      <label><input type="radio" name="work" value="自営業"> 
      自営業</label><br>
      <label><input type="radio" name="work" value="その他"> 
      その他</label><br>
      <input type="button" id="checkButton" value="チェック">
    </form>
    <div id="result">結果:</div>
  </body>
</html>
```

1 …ラジオボタンには「name」属性で名前を指定しておく

SOURCE CODE JavaScript main.js

```javascript
window.onload = function(){
  document.getElementById("checkButton").onclick = function(){
    var radioList = document.getElementsByName("work"); ―1
    var str = "選択されていません";
    for(var i=0; i<radioList.length; i++){ ―2
      if (radioList[i].checked) { ―3
        str = radioList[i].value + "が選択されています";
        break;
      }
    }
    document.getElementById("result").innerHTML = str;
  }
}
```

1 …「work」という名前を持つラジオボタンの情報を取得する

2 …ラジオボタンの数だけ繰り返す

3 …「checked」プロパティが「true」の場合は選択されているので選択項目を表示する

OnePoint 「getElementsByName()」で同じ名前のエレメントを取得する

「document.getElementsByName()」を使うと、ページ上にある同じ名前のエレメント(ラジオボタン)を取得することができます。これはラジオボタンで同じ名前(グループ名)を指定するため、このメソッドを使うことで特定のラジオボタングループの情報を取得することができます。取得したラジオボタンの「checked」プロパティを調べることで、どのラジオボタンがONになっているかを取得することができます。

SECTION 044 どのチェックボックスがONになっているか調べる

ここでは、どのチェックボックスがONになっているか調べ、ONになっている項目名をページ上に表示する方法について解説します。

チェックボックスをONにして［チェック］ボタンをクリックすると…

ONになっているチェックボックスの項目一覧が表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; 
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css" 
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>お使いのOSをお答えください</h1>
```

```
    <form action="./enq.cgi" method="get" id="mainForm"
    name="mainForm">
      <label><input type="checkbox" id="OS1" name="OS1"
      value="Mac">MacOS X</label><br>
      <label><input type="checkbox" id="OS2" name="OS2"
      value="Unix">Linux / Unix</label><br>
      <label><input type="checkbox" id="OS3" name="OS3"
      value="Win">Windows</label><br>
      <label><input type="checkbox" id="OS4" name="OS4"
      value="etc">その他</label><br>
      <input type="button" id="checkButton" value="チェック">
    </form>
    <div id="result">結果:</div>
  </body>
</html>
```

1 …チェックボックスにはID名を指定しておく

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("checkButton").onclick = function(){
    var checkList = ["OS1","OS2","OS3","OS4"];
    var str = "";
    for(var i=0; i<checkList.length; i++){
      var chkObj = document.getElementById(checkList[i]);
      if (chkObj.checked) {
        str += chkObj.value + "が選択されています<br>";
      }
    }
    if (!str) str = "何も選択されていません";
    document.getElementById("result").innerHTML = str;
  }
}
```

1 …調べるチェックボックスのID名を配列に入れておく
2 …チェックボックスの数だけ繰り返す
3 …「checked」プロパティが「true」の場合は選択されているので選択項目を追加表示する

OnePoint 調べる対象となるチェックボックスのID名を配列に入れておく

チェックボックスのON/OFFの状態は、「checked」プロパティで取得することができます。「checked」プロパティが「true」であればON、「false」であればOFFになります。複数のチェックボックスを調べる場合には、あらかじめ調べるチェックボックスのID名を配列内に入れておきます。配列の数だけ繰り返しチェックを行うことで、どのチェックボックスがONになっているか調べることができます。

SECTION

045 チェックボックスがONになったらテキストフィールドを表示する

ここでは、チェックボックスがONになっているときだけ、テキストフィールドを表示する方法について解説します。

JavaScript Sample - Windows Internet Explorer

アンケート

意見を書く

チェックをONにすると…

テキストフィールドが表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; 
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css" 
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
```

```
  <body>
    <h1>アンケート</h1>
    <form action="./enq.cgi" method="get" id="mainForm"
    name="mainForm">
      <label><input type="checkbox" id="checkCmt" name="checkCmt">
      意見を書く</label><br>
      <textarea id="uComment" name="uComment"></textarea> ―1
    </form>
  </body>
</html>
```

1 …チェックされた際に表示されるテキストフィールドにはID名を指定しておく(スタイルシートで非表示にしておく)

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("checkCmt").onclick = function(){
    if (this.checked) { ―1
      document.getElementById("uComment").style.display = "block";
    }else{
      document.getElementById("uComment").style.display = "";
    }
  }
}
```

1 …チェックボックスがクリックされたときに選択されているかどうかに応じてスタイルシートの「display」に表示方法を設定する

OnePoint テキストフィールドの表示/非表示は「display」プロパティを操作する

チェックボックスがONになった場合のみ、テキストフィールドを表示するには、「display」プロパティを操作します。チェックボックスがONになったら、テキストフィールドの「display」プロパティに「"block"」の文字を設定します。これでテキストフィールドが画面上に表示されます。逆にチェックがOFFになった場合には、「display」プロパティに「""」(空文字)を設定します。

Column ページやエレメントのアクセスブロック

ページ全体や特定のブロックやエレメントだけアクセスさせたくない(アクセスブロック)場合があります。そのような場合は、jQueryを利用した次のライブラリを利用すると便利です。これはページ全体をアクセスブロックするだけでなく、特定のブロックやエレメントだけをアクセスブロックすることができます。

URL http://malsup.com/jquery/block/

チェックボックスがONになったらテキストフィールドへの入力を許可する

ここでは、チェックボックスがONになっているときだけ、テキストフィールドへの入力を許可する方法について解説します。

SOURCE CODE　HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
```

```
  <body>
    <h1>ユーザー登録</h1>
    このWebサイト内の情報の転載を禁止します。<br>
    <form action="./enq.cgi" method="get" id="mainForm" ⤶
    name="mainForm">
      <label><input type="checkbox" id="checkAccept" ⤶
      name="checkAccept">同意する</label><br>
      メールアドレス:<input type="text" id="uComment" ⤶
      name="uComment" disabled>                          ─1
    </form>
  </body>
</html>
```

1 …チェックされた際に入力が許可されるテキストフィールドにはID名を指定し、また、「disabled」として入力不可にしておく

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("checkAccept").onclick = function(){
    document.getElementById("uMail").disabled = !this.checked; ─1
  }
}
```

1 …チェックボックスがクリックされたらテキストフィールドの「disabled」プロパティに「true」または「false」を設定する

OnePoint チェックボックスの状態を調べてテキストフィールドの「disabled」プロパティに設定する

チェックボックスにチェックが入った場合のみテキストフィールドへの入力を許可するには、「disabled」プロパティに「false」を設定します。

サンプルではチェックボックスの選択状態を反転させた状態を設定しています。チェックボックスがONの場合には「true」なので、これを反転させた「false」がテキストフィールドの「disabled」プロパティに設定され入力が可能な状態になります。チェックボックスがOFF(false)の場合には「true」がテキストフィールドの「disabled」プロパティに設定されることになります。

Column prototype.jsライブラリを利用した空欄チェック

prototype.jsライブラリには、フォームのエレメントの空欄チェックを行うメソッドが用意されています。空欄チェックは「Field.present()」で行い、パラメータにチェックしたいテキストフィールドのID名またはエレメントを指定します。パラメータは「,」(カンマ)で区切って複数指定することができます。「Field.present()」の戻り値が「true」であれば指定されたすべてのテキストフィールドは入力されていて、「false」であれば入力されていない(空欄があった)ことになります。あとは、戻り値を調べて処理を分ければよいでしょう。

SECTION 047 チェックボックスのON/OFFに応じてテキストフィールドを追加/削除する

ここでは、DOMで利用できるメソッドを使って、チェックボックスをONにしたらテキストフィールドを追加し、OFFにしたら削除する方法について解説します。

チェックをONにするとテキストフィールドが追加され、チェックをOFFにするとテキストフィールドが削除される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
```

```
  <body>
    JavaScriptを利用して、チェックボックスのON/OFFでテキストフィールドの追加と削除を行っています。
    <form action="./enq.cgi" method="get" id="mainForm"
    name="mainForm">
      <label><input type="checkbox" id="checkAccept"
      name="checkAccept">同意してメールアドレスを登録する</label><br>
      <div id="cFieId"></div> ―1
    </form>
  </body>
</html>
```

1 …チェックのON/OFFに応じて追加/削除するテキストフィールドの領域を用意する

SOURCE CODE JavaScript **main.js**

```
window.onload = function(){
  document.getElementById("checkAccept").onclick = function(){
    if (this.checked){
      var inputTag = document.createElement("input");
      inputTag.type = "text";
      inputTag.id = "uMail";
      inputTag.name = "uMail";
      document.getElementById("cField").appendChild(inputTag);
    }else{
      var fChild = document.getElementById("cField").firstChild;
      document.getElementById("cField").removeChild(fChild);
    }
  }
}
```

1 …チェックボックスがONの場合には<input>タグを生成し、ID名が「cField」の領域に追加する

2 …削除時にはID名が「cField」の領域の最初の子ノードを「removeChild()」を使って削除する

OnePoint 「appendChild()」「removeChild()」でテキストフィールドの追加/削除を行う

DOMでチェックボックスにチェックが入った場合のみテキストフィールドを追加するには、「appendChild()」を使います。「appendChild()」で追加する<input>タグは、「document.createElement()」で生成します。生成した後に必要なプロパティを設定しますが、必ず「type」プロパティから設定します。他のプロパティから設定するとInternet Explorerでは正しく反映されず、動作しないことがあります。

追加したテキストフィールドを削除するには、「removeChild()」を使います。追加されたテキストフィールドはサンプルでは「<div id="cField">」の最初の子ノードなので、「document.getElementById("cField").firstChild」としてアクセスすることができます。この子ノードを「removeChild()」のパラメータに指定することでテキストフィールドを削除することができます。

SECTION

048 セレクトメニューで選択された項目を取得する

ここでは、セレクトメニューで選択されたページ名(項目)に応じて、対応するページを表示する方法について解説します。

セレクトメニューを選択すると…

選択したページに移動する

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; 
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css" 
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
```

```
<body>
    下記のメニューで選択したページにジャンプします。<br>
    <form action="./jump.cgi" method="get" id="mainForm" 
    name="mainForm">
      <select id="jump"> ―1
        <option value="http://www.yahoo.co.jp/">Yahoo!</option>
        <option value="http://www.google.co.jp/">Google</option>
        <option value="http://www.openspc2.org/">OpenSpace</option>
      </select>
    </form>
  </body>
</html>
```

1 …セレクトメニューにID名を指定する

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("jump").onchange = function(){ ―1
    location.href = this.value; ―2
  }
}
```

1 …項目が選択された場合に処理するようにイベントハンドラを設定する
2 …選択項目の「value」プロパティを読み出し、ページのURLとして設定する

OnePoint 「this.value」で選択項目の内容を読み出す

セレクトメニューで項目が選択されると、「onChange」イベントが発生します。選択された項目の内容を読み出すには、セレクトメニューの「value」プロパティを読み出します。サンプルでは、「this.value」としてセレクトメニューで選択された項目の内容を読み出しています。読み出した内容を「location.href」に設定すると、指定されたページを表示することができます。

Column 入力フォームの特定のエレメントだけ取得する

prototype.jsライブラリには、入力フォームの特定のエレメントだけ取得するメソッドが用意されています。「Form.getInputs()」で最初のパラメータに入力フォームのID名、2番目のパラメータに取得したい種類を指定します。この種類は、<input>タグの「type」属性で指定されている値になります。「Form.getInputs()」と「each()」を組み合わせることでミスの少ないプログラムを作成することができるようになります。

049 セレクトメニュー項目を追加する

ここでは、ページが読み込まれたタイミングで、セレクトメニュー項目を追加する方法について解説します。

JavaScript Sample - Windows Internet Explorer

下記のメニューはJavaScriptによって追加されています。

Yahoo!
Yahoo!
Google
OpenSpace

Google - Windows Internet Explorer

ページが読み込まれたときにセレクトメニューの項目が追加される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
```

```
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    下記のメニューはJavaScriptによって追加されています。<br>
    <form action="./jump.cgi" method="get" id="mainForm"
    name="mainForm">
      <select id="jump"></select>  ―1
    </form>
  </body>
</html>
```

1 …セレクトメニューにID名を指定する

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  var selObj = document.getElementById("jump");
  var optionData = [
    { text:"Yahoo!", value:"http://www.yahoo.co.jp/"},
    { text:"Google", value:"http://www.google.co.jp/"},
    { text:"OpenSpace", value:"http://www.openspc2.org/"}   ―1
  ];
  for (var i=0; i<optionData.length; i++){
    selObj.options[i] =
    new Option(optionData[i].text, optionData[i].value);
  }
  selObj.onchange = function(){  ―2
    location.href = this.value;
  }
}
```

1 …追加するセレクトメニューの項目を連想配列(ハッシュ)として用意する
2 …「new Option()」として項目名と「value」プロパティを指定して生成する

OnePoint セレクトメニューで表示される項目を追加するには「new Option()」を使う

セレクトメニューで表示される項目を追加するには、「new Option()」として追加する項目名と「value」プロパティ値を設定します。複数の項目を追加する場合には、配列などに追加項目を入れておき繰り返し追加するようにします。また、項目を追加する際には、何番目の項目に追加するかを指定する必要があります。

SECTION 050 セレクトメニュー項目に応じて別のセレクトメニュー項目の内容を切り替える

ここでは、セレクトメニュー項目から選択した内容に応じて、もう1つのセレクトメニュー項目の内容を切り替える方法について解説します。

JavaScript Sample - Windows Internet Explorer

OSの調査

OSを選択してください / Windows / Mac OS

一方のセレクトメニューを選択すると…

もう片方の項目が状況に応じて変化する

Windows / 95〜Me / 2000/XP / Vista

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>OSの調査</h1>
    <form action="./enq.cgi" method="get" id="mainForm"
    name="mainForm">
      <select id="selMain"> ―1
        <option value="">OSを選択してください</option>
        <option value="Win">Windows</option>
        <option value="Mac">Mac OS</option>
      </select>
```

```
      <select id="selSub"></select> ―2
    </form>
  </body>
</html>
```

1 …セレクトメニューにID名を指定する

2 …内容が変更されるセレクトメニューのID名を指定する

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("selMain").onchange = function(){
    var selNo = this.selectedIndex; ―1
    var selObj = document.getElementById("selSub");
    var optionData = [[
      ],[
        { text:"95~Me", value:"95"},
        { text:"2000/XP", value:"2000"},
        { text:"Vista", value:"Vista"}
      ],[
        { text:"OS 9", value:"9"},
        { text:"OS X", value:"10"}
      ]
    ];
    while(selObj.hasChildNodes()) {
      selObj.removeChild(selObj.firstChild);
    }
    for (var i=0; i<optionData[selNo].length; i++){
      selObj.options[i] = new Option(optionData[selNo][i].text,
      optionData[selNo][i].value);
    }
  }
}
```

1 …選択されたセレクトメニューの値を読み出す

2 …追加する項目名と「value」プロパティの値を配列に用意しておく

3 …追加前には既存の項目を全て削除する

4 …選択されたセレクトメニューの値に応じて配列内容を読み出し、セレクトメニューの項目として追加する

OnePoint 選択状況に応じて配列に追加項目などを用意しておく

セレクトメニューで選択された項目に応じて、別のセレクトメニューの内容を変更するには、あらかじめ配列にセレクトメニューの項目を用意しておきます。選択された項目の選択番号に応じて配列内容を読み出し、「new Option()」でセレクトメニューの項目を追加します。ただし、選択項目の総数が変化するような場合には、必ず選択項目を削除してから追加するようにします。削除しない場合は、以前の項目が残ったままになります。

SECTION 051 ボタンをクリックするたびに異なる処理をランダムに行う

ここでは、ボタンをクリックするたびに異なる通貨単位で表示されるようにする方法について解説します。

当選金額

金額: 45687 実行

当選金額は 45687ドルです

当選金額は 45687円です

当選金額は 45687ポンドです

ボタンをクリックする度に異なる単位での金額が表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
```

```
    <body>
        <h1>当選金額</h1>
        <form action="./enq.cgi" method="get" id="mainForm"
        name="mainForm">
            金額:<input type="text" name="price" id="price" value="215">
            <input type="button" id="setButton" value="実行">
        </form>
        <div id="result"></div>
    </body>
</html>
```

SOURCE CODE　JavaScript　main.js

```
window.onload = function(){
    document.getElementById("setButton").onclick = function(){
        var n = Math.floor(Math.random() * rndFunc.length);  ─┐1
        rndFunc[n]();                                          ─┘
    }
}
var rndFunc = [
    function(){ ─2
        var pr = document.getElementById("price").value;
        document.getElementById("result").innerHTML = "当選金額は"+pr+"円です";
    },
    function(){ ─3
        var pr = document.getElementById("price").value;
        document.getElementById("result").innerHTML = "当選金額は"+pr+"ドルです";
    },
    function(){ ─4
        var pr = document.getElementById("price").value;
        document.getElementById("result").innerHTML = "当選金額は"+pr+"ポンドです";
    }
]
```

1 …求めた乱数値を配列の添字として関数「rndFunc」を呼び出す
2 …テキストフィールドに入力された値に「当選金額は～円です」の文字列を追加して表示する
3 …テキストフィールドに入力された値に「当選金額は～ドルです」の文字列を追加して表示する
4 …テキストフィールドに入力された値に「当選金額は～ポンドです」の文字列を追加して表示する

OnePoint 配列に実行する関数を入れる

ランダムに実行する関数を変更するには、あらかじめ配列に関数を定義しておきます。JavaScriptでは配列に関数を入れることもでき、実行する際にはサンプルのように「rndFunc[n]()」として呼び出すことができます。

SECTION 052 特定の条件に合うテキストフィールドに入力された値のみ加算する

ここでは、特定の条件に合うテキストフィールドに入力された値のみ加算し、合計を別のテキストフィールドに表示する方法について解説します。

計算ボタンをクリックすると…

商品の合計金額が表示される

SOURCE CODE　HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; 
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css" 
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <form action="./calc.cgi" method="get" id="mainForm" 
    name="mainForm">
      商品1:<input type="text" name="num1" id="num1" 
      value="1" size="3">個×
              <input type="text" name="price1" id="price1" 
              value="300">円<br>
      商品2:<input type="text" name="num2" id="num2" value="2" 
  1   size="3">個×
```

CHAPTER-03 フォーム

```
                <input type="text" name="price2" id="price2"
                value="100">円<br>
        商品3:<input type="text" name="num3" id="num3" size="3">個×
                <input type="text" name="price3" id="price3"
                value="150">円<br>
                <input type="text" name="total" id="total" >円<hr>
        特　記:<input type="text" name="userSpec" id="userSpec"><br>
        ご意見:<input type="text" name="userComment" id="userComment"><br>
        <input type="button" id="calcButton" value="計算">
      </form>
    </body>
</html>
```

1 …計算するテキストフィールドには「名前+番号」を指定する

SOURCE CODE　JavaScript　main.js

```
window.onload = function(){
  document.getElementById("calcButton").onclick = function(){
    var totalPrice = 0;
    for (var i=1; i<=3; i++){
  1── var num = parseInt(document.getElementById("num"+i).value) || 0;
  2── var prc = parseInt(document.getElementById("price"+i).value) || 0;
  3── totalPrice += num * prc;
    }
    document.getElementById("total").value = totalPrice;
  }
}
```

1 …テキストフィールド名が「num」と数値の1〜3の組み合わせのテキストフィールドの内容を読み出す(内容が入力されていない場合は0にする)

2 …テキストフィールド名が「price」と数値の1〜3の組み合わせのテキストフィールドの内容を読み出す(内容が入力されていない場合は0にする)

3 …合計金額(単価×個数)を求める

OnePoint 「名前+番号」の組み合わせでID名を指定する

特定の条件に合うテキストフィールドを読み出すには、いくつか方法があります。もっとも簡単な方法は、テキストフィールドの「name」属性または「id」属性で名前と番号の組み合わせを利用してアクセスする方法です。この方法であれば、「for()」を使って簡単に名前と番号の組み合わせのテキストフィールドを参照することができます。

また、スタイルシートのクラス名を指定する方法もあります。この場合は、正規表現の「match()」を使って対象となるクラスが適用されているかどうか調べて処理を行います。

SECTION

053 フォームの送信ボタンが押されたら処理を行う

ここでは、フォームの送信ボタン（例題では［登録］ボタン）が押されたら、送信前の処理を行う方法について解説します。

［登録］ボタンをクリックすると…

確認のダイアログが表示される

SOURCE CODE　HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>住所登録</h1>
    <form action="./regist.cgi" method="get" id="mainForm"
    name="mainForm">                                          ─ 1
      名前:<input type="text" name="userName" id="userName"><br>
      住所:<input type="text" name="userAddress" id="userAddress"><br>
      <input type="submit" value="登録">
    </form>
  </body>
</html>
```

1 …入力フォームにID名を指定しておく

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("mainForm").onsubmit = function(){ ―1
    return confirm("この内容で登録しますか?"); ―2
  }
}
```

1 …送信ボタンが押されると入力フォームの「submit」イベントが発生するのでイベントハンドラを設定する

2 …イベントハンドラからの戻り値が「true」であれば送信され、「false」であれば送信されない（確認ダイアログである「confirm()の戻り値は「true」か「false」であり、[OK]ボタンがクリックされれば送信、[キャンセル]ボタンがクリックされれば送信しないことになる）

OnePoint 入力フォームの「onsubmit」に送信時の処理を指定する

入力フォームの送信時に処理を行うには、「onsubmit」に送信前の処理を指定します。サンプルでは送信前に確認のダイアログを表示し、結果を「return」で返しています。「submit」イベントでは、イベントハンドラからの戻り値が「true」であれば送信、「false」であれば送信しないようになっています。サンプルのように、「confirm()」での確認ダイアログの戻り値を利用すると、[OK]ボタンがクリックされたときは送信され、[キャンセル]ボタンがクリックされたときには送信しないことになります。

Column クロージャ

JavaScriptでは、次のようにして関数内にさらに関数を定義することでクロージャを利用することができます。クロージャを利用することにより、グローバル変数などを利用したりすることなく、値を保持することができます。これにより、オブジェクト情報が隠蔽されるため、バグの軽減にもつながります。

```
window.onload = function(){
  aCount = xCounter(0);
  document.getElementById("aButton").onclick = function(){
    document.getElementById("dataValue").value = aCount();
  }
}
function xCounter(n){
  var cValue = n;  // 初期化
  var counter  = function(){
    cValue = cValue + 1;
    if (cValue > 9) cValue = 0;
    return cValue;
  }
  return counter;
}
```

054 フォームのリセットボタンが押されたら処理を行う

ここでは、フォームのリセットボタンが押されたら、リセットされる前に処理を行う方法について解説します。

JavaScript Sample - Windows Internet Explorer
住所登録
名前: 鈴木次郎
住所: 東京都港区
リセット 登録

リセットボタンをクリックすると…

確認のダイアログが表示され…

Windows Internet Explorer
内容をリセットしますか?
OK キャンセル

[OK]ボタンをクリックすると…

JavaScript Sample - Windows Internet Explorer
住所登録
名前:
住所:
リセット 登録

内容がリセットされる

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
```

```
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>住所登録</h1>
    <form action="./regist.cgi" method="get" id="mainForm"
    name="mainForm">   ―1
      名前:<input type="text" name="userName" id="userName"><br>
      住所:<input type="text" name="userAddress" id="userAddress"><br>
      <input type="reset" value="リセット">
      <input type="submit" value="登録">
    </form>
  </body>
</html>
```

1 …入力フォームにID名を指定しておく

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("mainForm").onreset = function(){ ―1
    return confirm("内容をリセットしますか?"); ―2
  }
}
```

1 …リセットボタンが押されると入力フォームの「reset」イベントが発生するのでイベントハンドラを設定する

2 …イベントハンドラからの戻り値が「true」であれば内容がリセットされ、「false」であれば送信されない(確認ダイアログである「confirm()」の戻り値は「true」か「false」であり、[OK]ボタンがクリックされれば内容をリセット、[キャンセル]ボタンがクリックされれば内容をリセットしないことになる)

OnePoint 入力フォームの「onreset」にリセット時の処理を指定する

入力フォームのリセットボタンがクリックされたときに処理を行うには、「onreset」にリセット前の処理を指定します。サンプルではリセット前に確認のダイアログを表示し、結果を「return」で返しています。「reset」イベントではイベントハンドラからの戻り値が「true」であれば送信、「false」であれば送信しないようになっています。サンプルのように「confirm()」での確認ダイアログの戻り値を利用すると、[OK]ボタンがクリックされたときには内容がリセットされ、[キャンセル]ボタンがクリックされたときには内容はリセットしないことになります。

Break Time

ステータスバーへの書き替えは許可なしにできない

以前はウィンドウのステータスバーへの書き込みは、次のサンプルのように自由に行うことができました。

```
setInterval("(function (){ window.status = new Date })()",
1000);
```

JavaScriptで一番最初に注目されたのも、ステータスバーにメッセージが流れるというプログラムでした。また、デバッグ時にも値を表示する際に便利な領域として利用されることもありました。

しかし、現在ではリンク先のURLが偽装されたりするため、ユーザーがステータスバーへの書き込みを許可しない限り文字列を表示できないようになっています。Firefox2とInternet Explorer7では、次のように設定を変更します。

● Firefox2の場合

❶ [ツール(T)]→[オプション(O)]を選択します。

❷「コンテンツ」のカテゴリの 詳細設定(V)... ボタンをクリックします。

❸ [ステータスバーのテキストを変更する(C)]をONにして OK ボタンをクリックします。

◉Firefox2

● Internet Explorer 7の場合

❶ [ツール(O)]→[インターネットオプション(O)]を選択します。

❷「セキュリティ」タブの レベルのカスタマイズ(C)... ボタンをクリックします。

❸ [スクリプトでのステータスバーの更新を許可する]で[有効にする]をONにして OK ボタンをクリックします。

◉Internet Explorer7の場合

CHAPTER - 04

画像

055 画像を制御する方法について

画像の制御方法

JavaScriptでは、画像を制御(参照や設定)する方法が複数あり、目的に応じて使い分けます。

▶「images」配列(連想配列)を利用する方法

「images」配列を利用する方法は、古いブラウザでも新しいブラウザでも問題なく利用できます。「images」配列は連想配列でもあるため、参照番号または<img>タグの「name」属性で画像を指定することができます。配列の参照番号は、HTML文書内で<img>タグが出現した順番に番号が割り振られます。

▶DOMを利用する方法

DOMが利用できるブラウザであれば、<img>タグにID名を指定し、「document.getElementById()」を使って直接制御することができます。これは<img>タグだけではなく他の一般的なタグのアクセスと同様です。また、「getElementsByTagName()」を使って特定の領域の<img>タグだけを抽出することもできます。他に、ノードにアクセスすることで画像へのアクセスが可能です。

▶ライブラリを利用する方法

prototype.jsライブラリやjQueryなどのライブラリを利用すると、スタイルシートのクラス名やHTMLタグの階層構造を指定して制御することができます。prototype.jsライブラリでは「$$()」、jQueryでは「$()」を使って、パラメータにクラス名/階層構造を指定します。

サンプルでは、それぞれ異なる方法を使って、同一の画像「images/01.jpg」を呼び出している。[画像を変更]ボタンをクリックすると一斉に「images/02.jpg」に切り替わる

SOURCE CODE　HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; ↵
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css" ↵
    media="all">
    <script type="text/javascript" src="prototype.js"></script>
    <script type="text/javascript" src="main.js"></script>
  </head>
```

```html
  <body>
    JavaScriptによって指定した画像にアクセスし、ボタンをクリックすることで画像を入れ替えます。
    <form action="./check.cgi" method="get" name="mainForm">
      <input type="button" id="setButton" value="画像を変更">
    </form>
    <div>
      <img src="images/01.jpg" name="photo1">
      images配列でアクセス(参照番号を指定)<br>
      <img src="images/01.jpg" name="photo2">
      images配列でアクセス(name属性を指定)<br>
      <img src="images/01.jpg" id="photo">
      document.getElementById()でアクセス(id属性を指定)<br>
      <img src="images/01.jpg" class="HDV">
      prototype.jsでアクセス(class属性を指定)<br>
    </div>
    <div id="photoAlbum">
      <img src="images/01.jpg"> getElementsByTagName()でアクセス<br>
      <img src="images/01.jpg" class="HDV">
      prototype.jsでアクセス(class属性を指定)<br>
    </div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```javascript
window.onload = function(){
  document.getElementById("setButton").onclick = function(){
    var imgURL = "images/02.jpg";
    document.images[0].src = imgURL; 1
    document.images["photo2"].src = imgURL; 2
    document.getElementById("photo").src = imgURL; 3
    $$(".HDV").each(function(obj){
      obj.src = "images/02.jpg";
    }); 4
    var root = document.getElementById("photoAlbum");
    var childImg = root.getElementsByTagName("img");
    childImg[0].src = imgURL; 5
  }
}
```

1 …「images」配列を利用して最初の画像のURLを指定している

2 …「images」配列で画像の名前をキーにして画像のURLを指定している

3 …DOMを使い、ID名が「photo」の画像のURLを指定している

4 …prototype.jsライブラリを使い、スタイルシートのクラス名が「HDV」の画像のURLを指定している

5 …DOMを使い、ID名が「photoAlbum」の<div>タグ内に含まれる<img>タグの最初の画像のURLを指定している

SECTION 056

一定時間ごとに画像を自動的に切り替えて表示する

ここでは、一定時間ごとに次々と画像を入れ替える方法について解説します。

JavaScript Sample - Windows Internet Explorer

私の好きな風景

最近、特に気に入っている風景の写真です。

2秒ごとに画像が入れ替わる

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
```

```
  <body>
    <h1>私の好きな風景</h1>
    最近、特に気に入っている風景の写真です。
    <div class="photoAlbum">
      <img src="images/DSC0001.jpg" id="photo"> ―1
    </div>
  </body>
</html>
```

1 …入れ替える画像にid属性でID名を指定する

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  setInterval("album.swapImage()", 2000); ―1
}
var album = {
  imageURL : [
              "images/DSC0001.jpg",
              "images/DSC0042.jpg",
              "images/DSC0103.jpg",   ―2
              "images/DSC0154.jpg",
              "images/DSC0280.jpg"
              ],
  swapImage : function(){
    this.count++;
    if (this.count >= this.imageURL.length) this.count = 0;  ―3
4― document.getElementById("photo").src = this.imageURL[this.count];
  },
  count : 0 ―5
};
```

1 …タイマーを使って2秒ごとに画像を入れ替える処理(「album.swapImage()」)を呼び出す
2 …入れ替える画像のURLを配列として用意する
3 …画像を表示する位置を示すカウンタを増やし、配列数を超えたら0に戻す
4 …カウンタに対応した画像を入れ替える
5 …カウンタを初期化する(0にする)

OnePoint タイマーを使って一定時間ごとに画像URLを入れ替える

一定時間ごと画像を入れ替えるには、「setInterval」タイマーを使って画像のURLを指定します。入れ替える画像のURLは、あらかじめ配列内に指定しておきます。この配列の何番目の画像を表示するかを示すカウンタを用意します。このカウンタを1ずつ増やして行くことで画像を入れ替えることができます。カウンタが配列の要素数を超えたら0に戻すことで延々と繰り返すことができます。

SECTION 057 マウスカーソルが重なったら画像を切り替える

ここでは、マウスカーソルが重なったら画像を切り替える方法について解説します。

マウスカーソルを重ねると
タブの画像が切り替わる

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>C&amp;R自動車へようこそ!</h1>
    <div id="mainMenu">―1
      <img src="images/menu1.gif" width="84" height="24">
      <img src="images/menu2.gif" width="84" height="24">
```

```
      <img src="images/menu3.gif" width="84" height="24">
    </div>
    <img src="images/pic001.jpg"  width="250" height="150">
  </body>
</html>
```

1 …入れ替える画像を囲む領域を<div>タグで設定する

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  var rootNode = document.getElementById("mainMenu");   ─ 1
  var imgTag = rootNode.getElementsByTagName("img");
  for (var i=0; i<imgTag.length; i++){   ─ 2
    imgTag[i].onmouseover = function(){
      this.src = this.src.split(".gif")[0]+"_over.gif";   ─ 3
    }
    imgTag[i].onmouseout = function(){
      this.src = this.src.split("_over.gif")[0]+".gif";   ─ 4
    }
  }
}
```

1 …ID名が「mainMenu」の<div>タグを基準として<img>タグを抽出する
2 …<img>タグの数だけ繰り返し、マウスオーバー時、マウスアウト時の処理を追加する
3 …マウスが重なったら画像名に「_over.gif」を追加したファイル名に入れ替える
4 …マウスが離れたら「_over.gif」を削除したファイル名に入れ替える

OnePoint 「onmouseover」と「onmouseout」に画像を入れ替える処理を設定する

マウスが画像の上に重なったら画像を入れ替える、離れたら元の画像に戻すといったロールオーバーの処理は、「onmouseover」イベントと「omouseout」イベントを利用します。従来はHTMLタグの<a>または<img>タグ内にイベントを記述していますが、DOMが利用できるブラウザではサンプルのように特定の領域に限定して画像のロールオーバーを行うことができます。画像のロールオーバーは<img>タグ自身の「src」プロパティを入れ替えるため、「this.src」として指定し表示する画像URLを指定します。

サンプルでは元の画像ファイル名に「_over」を追加した画像をロールオーバー画像として設定しているため、次のように画像ファイル名が設定されていないと動作しません。

ファイル名	内　容
menu1.gif	タブ項目1の標準画像
menu1_over.gif	タブ項目1のロールオーバー画像
menu2.gif	タブ項目2の標準画像
menu2_over.gif	タブ項目2のロールオーバー画像
menu3.gif	タブ項目3の標準画像
menu3_over.gif	タブ項目3のロールオーバー画像

058 マウスカーソルが重なったら複数の画像を切り替える

ここでは、マウスカーソルが重なったときに複数の画像をまとめて切り替える方法について解説します。

リンク文字にマウスカーソルを重ねると…

3枚の画像が切り替わる

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; ↩
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css" ↩
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>我が家の愛犬ギャラリー</h1>
    <div id="photoAlbum"> ―1
      <a href="#" id="changeButton1">ギャラリー(1)</a> |    ┐
      <a href="#" id="changeButton2">ギャラリー(2)</a><br> ┘2
      <img src="images/DSC0001.jpg" width="240" height="135">
```

```
        <img src="images/DSC0042.jpg" width="240" height="135">
        <img src="images/DSC0103.jpg" width="240" height="135">
      </div>
   </body>
</html>
```

1 …入れ替える画像を示す領域を設定する

2 …イベントを設定する<a>タグにID名を指定する

SOURCE CODE　JavaScript **main.js**

```
window.onload = function(){
  document.getElementById("changeButton1").onmouseover =
  function() {
    album3.change(["images/DSC0001.jpg","images/DSC0042.jpg",
    "images/DSC0103.jpg"]);
  }
  document.getElementById("changeButton2").onmouseover =
  function() {
    album3.change(["images/DSC0154.jpg","images/DSC0280.jpg",
    "images/DSC1005.jpg"]);
  }
}
var album3 = {
  change : function(imageArray){
    var imgTag = document.getElementById("photoAlbum").
    getElementsByTagName("img");
    for (var i=0; i<imgTag.length; i++){
      imgTag[i].src = imageArray[i];
    }
  }
}
```

1 …リンク文字にマウスが重なった時に入れ替える画像URLを処理するメソッドに渡す

2 …ID名が「photoAlbum」内にある<img>タグを抽出する

3 …<img>タグの数だけ繰り返し画像URLを設定する

OnePoint 入れ替える画像URLは配列で渡す

マウスオーバーで複数の画像を入れ替えるには、入れ替える画像のURLを配列として用意するか、そのまま入れ替えを行う関数やメソッドに渡します。サンプルでは、「album3.change()」に配列として渡しています。実際に画像を入れ替える処理を行う「album3.change()」では渡された配列を参照し、画像URLとして設定するだけなので短いコードで済みます。

サンプルのように入れ替える画像が特定の領域に含まれるか連続している場合は簡単ですが、そうでない場合にはID名またはスタイルシートのクラス名を参照して画像を入れ替える方法を使います。

SECTION 059

画像をクリックするたびに次々と画像を切り替えて表示する

ここでは、クリック操作で次々と画像を切り替えて表示する方法について解説します。

JavaScript Sample - Windows Internet Explorer
C:¥JavaScript-Sa
Live Search
JavaScript Sample
ネコの写真館
近所のネコを撮ってみました。
※写真をクリックすると、次の写真が表示されます。
コンピュータ | 保護モード: 無効
100%

画像をクリックすると次の画像が表示される

SOURCE CODE　HTML ▶ index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; 
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css" 
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
```

```
  <body>
    <h1>ネコの写真館</h1>
    近所のネコを撮ってみました。<br>
    ※写真をクリックすると、次の写真が表示されます。
    <div class="photoAlbum">
      <img src="images/DSC0001.jpg" id="photo"> ―1
    </div>
  </body>
</html>
```

1 …クリックで入れ替える画像にID名を指定する

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("photo").onclick = album5.nextImage; ―1
}
var album5 = {
  imageURL : [
              "images/DSC0001.jpg",
              "images/DSC0042.jpg",
              "images/DSC0103.jpg",
              "images/DSC0154.jpg",
              "images/DSC0280.jpg"
              ],
  nextImage : function(){
    album5.count++;
    if (album5.count >= album5.imageURL.length) album5.count = 0; ―2
    document.getElementById("photo").src =
     album5.imageURL[album5.count]; ―3
  },
  count : 0 ―4
};
```

1 …画像にクリックイベント発生時の処理を設定する

2 …表示する画像を示すカウンタを増やし、枚数以上になったら0に戻す

3 …あらかじめ定義してある配列から画像URLを読み出し設定する

4 …カウンタを初期化する

OnePoint カウンタを用意して画像URLを設定する

クリックで画像を次々と表示していくには、あらかじめ配列に画像URLを定義しておきます。何番目の画像を表示するかを示すカウンタを用意し、クリックされたら1ずつ増やしていきます。カウンタが配列の要素数を超えたら0に戻します。これによりクリックし続けた場合、最初の画像にもどって延々と表示されるようになります。

SECTION 060

クリックしたときに画像のサイズを変更する

ここでは、クリックするたびに画像のサイズを大きくしたり、小さくしたりと、交互に変更する方法について解説します。

画像をクリックするたびに
画像のサイズが切り替わる

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
```

```
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>桜写真集</h1>
    <div id="photoAlbum">
      <img src="images/DSC0001.jpg" class="photoSmall frame">
      <img src="images/DSC0103.jpg" class="photoLarge frame">
      <img src="images/DSC0280.jpg" class="photoSmall frame">
    </div>
  </body>
</html>
```

1 …クリックでサイズを切り替える画像にスタイルシートクラスを指定する

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  var root = document.getElementById("photoAlbum");
  var imgTag = root.getElementsByTagName("img");
  for (var i=0; i<imgTag.length; i++){
    imgTag[i].onclick = function(){
      var cName = this.className;
      if (cName.match(/photoSmall/)) {
        this.className = cName.replace(/photoSmall/, "photoLarge");
        return;
      }
      if (cName.match(/photoLarge/)) {
        this.className = cName.replace(/photoLarge/, "photoSmall");
        return;
      }
    }
  }
}
```

1 …画像の数だけ繰り返す

2 …画像がクリックされたときのイベントを設定する

3 …現在のスタイルシートのクラス名を読み出す

4 …スタイルシートのクラス名に「photoSmall」が含まれている場合は正規表現を使って「photoLarge」に置換する

5 …スタイルシートのクラス名に「photoLarge」が含まれている場合は正規表現を使って「photoSmall」に置換する

SOURCE CODE CSS main.css

```
body {
  background-color:#fff;
  color:#333;
  font-size:80%;
  line-height:1.8;
}
h1 {
  font-size:14pt;
  color:#777;
  padding:0;
  margin:0;
}
.photoLarge {
  width:240px;
  height:135px;
}
.photoSmall {
  width:48px;
  height:27px;
}
.frame {
  border:1px #338 solid;
}
```

1 …写真を大きくしたときのピクセル数を指定する

2 …写真を小さくしたときのピクセル数を指定する

OnePoint あらかじめサイズを示すスタイルシートクラスを用意し入れ替える

クリックで画像のサイズを変更するには、あらかじめスタイルシートで小さいサイズのクラスと大きいサイズのクラスを定義しておきます。現在適用されているスタイルシートのクラスを「className」プロパティから読み出し正規表現を使って置換します。

直接スクリプトで画像のサイズを示すスタイルシートの「width」プロパティ、「height」プロパティを操作することもできます。ただし、文書構造と装飾、動作を分離する方が好ましいため、サンプルのようにスタイルシートのクラスを定義して切り替える方がベストです。この方法であればサイズを変更する場合もスクリプトを変更する必要がありません。

SECTION 061

ブラウザ上の画像を自在にドラッグする

ここでは、Yahoo!が提供しているYahoo UIライブラリを利用して、画像をドラッグする方法について解説します。

JavaScript Sample - Windows Internet Explorer

ピンナップ

※画像をドラッグして好きなところに配置してください。

画像をドラッグして好きな位置に移動することができる

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
```

```
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="yahoo.js"></script>
    <script type="text/javascript" src="dom.js"></script>
    <script type="text/javascript" src="event.js"></script>
    <script type="text/javascript" src="dragdrop.js"></script>
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>ピンナップ</h1>
    ※画像をドラッグして好きなところに配置してください。
    <div id="photoAlbum">
      <img src="images/DSC0001.jpg" id="photo1">
      <img src="images/DSC0103.jpg" id="photo2">
      <img src="images/DSC0280.jpg" id="photo3">
    </div>
  </body>
</html>
```

1 …ライブラリに必要なファイルを読み込ませる
2 …ドラッグする画像を用意する(このときに必ずID名を指定する)

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  var root = document.getElementById("photoAlbum");
  var imgTag = root.getElementsByTagName("img");
  for (var i=0; i<imgTag.length; i++){
    new YAHOO.util.DD(imgTag[i].id);
  }
}
```

1 …ドラッグ対象にする画像の数だけドラッグできるように繰り返す
2 …「YAHOO.util.DD()」のパラメータにドラッグ対象にする画像のIDを渡す

OnePoint ドラッグ処理は画像にID名を指定して「YAHOO.util.DD()」を使って設定する

Yahoo UIライブラリを利用すると手軽にドラッグ処理を行うことができます。ドラッグ対象となる画像やエレメントには必ずID名を指定しておきます。このID名を「YAHOO.util.DD()」のパラメータに設定することでドラッグすることが可能になります。

062 マウスカーソルが重なったら画像が浮き出るようにする

ここでは、script.aculo.usライブラリを利用して、半透明の画像の上にマウスカーソルを重ねたときだけ、画像の色が鮮明になるように表示する方法について解説します。

マウスカーソルを乗せると画像の不透明度が変化し鮮明になり、マウスカーソルを離すとやや透明になる

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css" media="all">
    <script type="text/javascript" src="prototype.js"></script>
    <script type="text/javascript" src="scriptaculous.js"></script>
```
1

```
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>ミニ美術館</h1>
    <div id="picGallery">
      <img src="images/pic01.jpg">
      <img src="images/pic02.jpg">
      <img src="images/pic03.jpg">
    </div>
  </body>
</html>
```

1 …script.aculo.usを利用するにはprototype.jsライブラリが必要なので両方とも読み込ませておく

SOURCE CODE JavaScript **main.js**

```
window.onload = function(){
  var root = document.getElementById("picGallery");
  var imgTag = root.getElementsByTagName("img");
  for (var i=0; i<imgTag.length; i++){
    imgTag[i].onmouseover = function(){  —1
  2— new Effect.Opacity(this, { from:0.5, to:1.0, duration:0.5 });
    }
    imgTag[i].onmouseout = function(){ —3
  4— new Effect.Opacity(this, { from:1.0, to:0.5, duration:0.5 });
    }
  }
}
```

1 …マウスが画像に重なった時の処理を設定する
2 …「Effect.Opacity()」で変化する不透明度の値と時間(秒)を設定する
3 …マウスが画像から離れた時の処理を設定する
4 …「Effect.Opacity()」で変化する不透明度の値と時間(秒)を設定する

OnePoint 「Effect.Opacity()」で不透明度の変化を設定する

script.aculo.usライブラリを利用すると、手軽に不透明度の処理を行うことができます。script.aculo.usライブラリはprototype.jsライブラリが必要なので、最初にprototype.sjライブラリを読み込ませます。マウスカーソルが重なったときには「onmouseover」イベントが発生するので、発生時の処理を指定しておきます。

不透明度を変化させるには、「Effect.Opacity()」を使います。最初のパラメータには不透明度を変化させるオブジェクトまたはID名を指定します。サンプルでは「this」を指定していますが、これは画像自身を示すことになります。「Effect.Opacity()」の2番目のオプションでは、開始不透明度と終了不透明度、そして変化する時間を秒数で指定します。

SECTION 063

スライドのように画像を表示する

ここでは、Lightboxライブラリを利用して、クリックした画像があたかもスライドのように表示する方法について解説します。

画像をクリックすると…

画像が画面上に表示され、周りは半透明グレーになる

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; 
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css" 
    media="all">
    <link rel="stylesheet" type="text/css"                    ―1
    href="lightbox.css" media="all">
    <script type="text/javascript" src="prototype.js"></script>   ―2
    <script type="text/javascript" 
    src="scriptaculous.js?load=effects"></script>
    <script type="text/javascript" src="lightbox.js"></script>
  </head>
  <body>                                                          ―3
    <h1>春のチューリップ展</h1>
    <a href="photo/01.jpg" rel="lightbox[album]" title="一重咲き">
      <img src="thumb/01.jpg">
    </a>
    <a href="photo/02.jpg" rel="lightbox[album]" title="トライアンフ系">
      <img src="thumb/02.jpg">
    </a>
    <a href="photo/03.jpg" rel="lightbox[album]" title="ダーウィン">
      <img src="thumb/03.jpg">
    </a>
  </body>
</html>
```

1 …Lightboxが使用するスタイルシートファイル(lightbox.css)を読み込む

2 …Lightboxはprototype.js、script.aculo.usに依存するので読み込ませる

3 …Lightboxを適用する部分には「rel」属性で「lightbox」を指定する

OnePoint Lightboxでスライド表示させるには「rel」属性に「lightbox」を指定する

Lightboxを利用するにはprototype.jsライブラリとscript.aculo.usライブラリが必要です。lightbox.jsを読み込ませる前に、この2つを順番に読み込ませておきます。Lightboxでスライド表示させたい画像を<a>タグで囲み、「rel」属性を指定します。「rel="lightbox"」とした場合は単独での表示になり、「rel="lightbox[グループ名]"」のようにグループ名を指定した場合は、一連のグループとして表示されます。また、画像のキャプションを表示するには<a>タグの「title」属性に設定します。

LightboxはHTML文書内に記述されている<a>タグの「rel」属性を読み取って自動的に設定するため、JavScriptプログラムを記述する必要はありません。

064 サムネールをクリックしたときに画像を拡大して表示する

ここでは、Highslideライブラリを利用して、クリックしたときにサムネール画像が拡大表示されるようにする方法について解説します。

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; 
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css" 
    media="all">
    <script type="text/javascript" src="highslide.js"></script> ―1
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>My Car Gallery</h1>
    <div id="photoAlbum">
      <div id="highslide-container"></div> ―2
      <a href="photo/01.jpg" id="thumb1" class="highslide"> ―3
        <img src="thumb/01.jpg"></a>
      <div class="highslide-caption" id="caption-for-thumb1"> ┐
        フロント                                              ├4
      </div>                                                  ┘
      <a href="photo/02.jpg" id="thumb2" class="highslide">
        <img src="thumb/02.jpg"></a>
      <div class="highslide-caption" id="caption-for-thumb2">
        サイド
      </div>
      <a href="photo/03.jpg" id="thumb3" class="highslide">
        <img src="thumb/03.jpg"></a>
      <div class="highslide-caption" id="caption-for-thumb3">
        ステアリング
      </div>
    </div>
  </body>
</html>
```

1 …Highslideライブラリを読み込ませる

2 …拡大画像を入れるためのコンテナを指定する

3 …Highslideの処理対象であることを示すためスタイルシートクラス名に「highslide」を指定し、また、キャプションと対応させるためにid名を指定する

4 …キャプションを指定する（ID名は「caption-for-」とHighslideの処理対象で指定したID名を付加した値になる）

SOURCE CODE JavaScript main.js

```
hs.graphicsDir = "./graphics/"; ―1
hs.outlineType = "rounded-white"; ―2
window.onload = function(){
  var root = document.getElementById("photoAlbum");
  var pObj = root.getElementsByTagName("a");
  for(var i=0; i<pObj.length; i++){
    pObj[i].onclick = function(){   ┐
      return hs.expand(this);       ┘3
    }
  }
  hs.preloadImages(); ―4
}
```

1 …Highslideで使用する画像のあるフォルダへのパスを指定する

2 …拡大表示された際の画像の枠の形状を指定する

3 …拡大表示された画像がクリックされたら元に戻るようにするための処理を追加する

4 …画像のプレロード処理を行う

OnePoint Highslideが利用する各種スタイルシートとID名を指定する

Highslideライブラリを利用するには、highslide.jsファイルを読み込ませます。その後、画像フォルダのパスやクリック時の設定を行います。拡大表示された画像がクリックされたら元のサムネール画像に戻るようにするには「return hs.expand(this)」を指定します。表示される画像の枠は、「hs.outlineType」プロパティに下表の値を指定します。

パラメータ	内　容
null	枠なし
outer-glow	発光
rounded-white	角丸四角形(白)
drop-shadow	影付き
beveled	ベベル

HTML文書では拡大画像を表示するためのコンテナ(<div id="highslide-container"></div>)を用意します。また、キャプションやクリック時に拡大される画像などには、特定のスタイルシートのクラス名とID名を指定します。キャプションを表示する場合には、<a>タグのID名と「caption-for-」を連結した値を<div>タグのID名として指定します。

なお、個人利用の場合は無料ですが、商用利用の場合には使用料を支払う必要があります。詳しくは、次のサイトを参照してください。

URL http://vikjavev.no/highslide/

Break Time

連想配列のキーに数値を使ってはいけない

JavaScriptの連想配列(ハッシュ)には、さまざまなキーを指定することができます。ただし、連想配列のキーに数値と解釈できる文字列を指定してはいけません。これは連想配列のキーが数値に変換され処理されてしまうためです。

次のサンプルでは、「C and R 研究所」と表示されるはずですが、予想した動作とは違い、「研究所」の文字が2行に渡って表示されてしまいます。つまり「"1"」は数値の「1」と解釈されてしまうのです。HTMLタグを操作する場合に「name」属性で数値だけの名前を指定してはいけないというのもこのような解釈が行われ予想と違う結果になってしまうためです。

連想配列は便利ですが連想配列のキーには整数値と解釈できる文字列は指定しないように注意してください。

```
a = new Array();
a[1] = "C and R";
a["1"] = "研究所";
document.getElementById("result").innerHTML = a[1]+"<br>";
document.getElementById("result").innerHTML += a["1"]+"<br>";
```

連想配列のキーに数値と
解釈できる文字列を指定
すると正しく実行されない

CHAPTER - 05

ダイアログ/ウィンドウ

065 アラートダイアログを開く

ここでは、アラートダイアログを開き、リンク先のURLを表示する方法について解説します。

リンクをクリックするとリンク先のURLをアラートダイアログに表示する

SOURCE CODE　HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>リンク集</h1>
    <ul id="displayAlert">
      <li><a href="http://www.yahoo.co.jp/">Yahoo! Japanへ</a></li>
      <li><a href="http://www.google.co.jp/">Googleへ</a></li>
      <li><a href="http://www.openspc2.org/">OpenSpaceへ</a></li>
    </ul>
    <ul>
      <li><a href="etc.html">その他のページへ</a></li>
      <li><a href="top.html">トップページへ</a></li>
    </ul>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  var rootNode = document.getElementById("displayAlert");
  var linkTag = rootNode.getElementsByTagName("a"); —1
  for (var i=0; i<linkTag.length; i++){
    linkTag[i].onclick = function(){ —2
      alert(this.href+"のページに移動します"); —3
    }
  }
}
```

1 …<a>タグの情報を読み出す
2 …リンクがクリックされたときの処理を設定する
3 …アラートダイアログにリンク先URLを表示する

OnePoint ダイアログを表示するには「alert()」を使う

利用者に注意などを促したい場合には、「alert()」を使ってダイアログを表示させることができます。「alert()」のパラメータに表示したい文字列や数値、変数、オブジェクトなどを指定します。

サンプルでは、特定の領域のリンクのみアラートを表示するようにしています。これは特定のID名を持つ<ul>タグで囲まれた範囲から「getElementsByTagName()」を使って<a>タグを抽出します。抽出した<a>タグにクリックイベントを設定することで各種処理を行わせることができます。

Column 古いブラウザでは簡単なプログラムでさえもわずかなコードの違いで動作しないことがある

2007年現在、JavaScriptでプログラムを作成する場合、対象とするブラウザはInternet Explorer 6以上、Firefox、Safari2、Opera8以上が多いでしょう。しかし、これよりも古いブラウザではちょっとしたことでもエラーとなり動作しないことがあります。

たとえば、Mac版Internet Explorer5では、読み込むスクリプトファイルのプログラムコードの前後に全角空白が入っているとエラーになります。全角空白が入っている場合にはSafari2でもエラーとなり、動作しません。

他にも文字コードをUTF-8などのユニコードにすると、正しく動作しない場合もあります。スタイルシートへの対応も異なるため、本書のようにHTMLファイル、スタイルシートファイル、スクリプトファイルを完全に分離すると動作しないこともあります。古いブラウザへの対応は過去に記述された数多くのコードを参考にするのがよいかもしれません。

●正常に動作しない場合の例（alert()の前に全角空白が入っている）

```
window.onload = function(){
　alert("Sample");
}
```

全角空白が入力されている

066 確認のダイアログを開く

ここでは、ボタンをクリックしたタイミングで確認のダイアログを開く方法について解説します。

JavaScript Sample - Windows Internet Explorer
C:¥JavaScript-Sample¥05_wi
Live Search
JavaScript Sample
ページ(P)
認証システム
認証コード: 456789
送信 リセット
ページ
コンピュータ | 保護モード: 無効
100%

[リセット]ボタンを
クリックすると…

確認のダイアログが
表示され…

Windows Internet Explorer
本当に消しますか？
OK キャンセル

[OK]ボタンをクリック
すると…

JavaScript Sample - Windows Internet Explorer
C:¥JavaScript-Sample¥05_wi
Live Search
JavaScript Sample
ページ(P)
認証システム
認証コード:
送信 リセット
ページ
コンピュータ | 保護モード: 無効
100%

入力した内容がリセットされる

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; 
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css" 
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>認証システム</h1>
    <form action="./regist.cgi" method="get" id="mainForm" 
    name="mainForm">
      認証コード:<input type="text" name="chkCode" id="chkCode" 
      value=""><br>
      <input type="submit" value="送信">
      <input type="reset" value="リセット">
    </form>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("mainForm").onreset = function(){ —1
    return confirm("本当に消しますか?"); —2
  }
}
```

1 …リセットボタンが押された時のイベント処理を設定する

2 …確認のダイアログを表示する

OnePoint 確認ダイアログを表示するには「confirm()」を使う

二者択一の選択を行わせたい場合には、「confirm()」を使います。特に念のため確認を促す場合に有効です。「confirm()」のパラメータに表示したい文字列を指定します。「confirm()」は、[OK]ボタンがクリックされた場合には「true」、[キャンセル]ボタンがクリックされた場合には「false」を返します。

サンプルでは「confirm()」の戻り値を「return」で呼び出し元に返していますが、リセットイベントが発生した際の戻り値が「true」の場合は内容がリセットされ、「false」の場合は内容はリセットされず、そのままになります。

067 文字入力ダイアログを開く

ここでは、文字入力ダイアログを開き、そこに入力した内容をページ上に表示する方法について解説します。

Explorer ユーザー プロンプト
スクリプト プロンプト:
お名前は？
OK
キャンセル
お名前を入力してください

ページが読み込まれると入力ダイアログが表示され…

Explorer ユーザー プロンプト
スクリプト プロンプト:
お名前は？
OK
キャンセル
鈴木一郎

文字を入力して[OK]ボタンをクリックすると…

JavaScript Sample - Windows Internet Explorer
C:¥JavaScript-Sai
Live Search
JavaScript Sample
Welcome!
ようこそ鈴木一郎さん
コンピュータ | 保護モード: 無効
100%

入力された文字がページ上に表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; 
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css" 
    media="all">
```

```html
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>Welcome!</h1>
    <div id="message"></div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```javascript
window.onload = function(){
  var userName = prompt("お名前は?", "お名前を入力してください"); —1
  if (!userName) userName = "名無 権兵衛"; —2
  document.getElementById("message").innerHTML =
  "ようこそ"+userName+"さん";                                    —3
}
```

1 …入力ダイアログを表示する
2 …名前が入力されなかったか[キャンセル]ボタンが押されたときの処理を指定する
3 …入力された名前をページ上に表示する

OnePoint 文字入力ダイアログを表示するには「prompt()」を使う

ダイアログを表示して文字入力を行うには、「prompt()」を使います。入力フォームでの文字入力と異なり、「prompt()」では文字を入力するかキャンセルされない限り、次の処理に移ることができません。強制的に何か入力させる場合に「prompt()」は適しています。

「prompt()」のパラメータは2つあり、最初が表示するメッセージ、2番目があらかじめ入力済みのデータ(デフォルト文字列)になります。文字列が入力されて[OK]ボタンがクリックされた場合、「prompt()」は入力された文字を返します。[キャンセル]ボタンがクリックされた場合には「prompt()」は「null」を返します。

◉[キャンセル]ボタンをクリックした場合

JavaScript Sample - Windows Internet Explorer
C:¥JavaScript-Sa
Live Search
JavaScript Sample
Welcome!
ようこそ名無 権兵衛さん
コンピュータ | 保護モード: 無効
100%

[キャンセル]ボタンがクリックされた場合は「名無 権兵衛」と表示される

068 モーダルダイアログを開く

ここでは、モーダルダイアログを開き、そこに入力した内容を親ページに表示する方法について解説します。

※このサンプルは、Internet Explorerのみで動作します。

JavaScript Sample -- Web ページ ダイアログ

名前を入れてください

鈴木一郎 決定

コンピュータ | 保護モード: 無効

ページが読み込まれるとモーダルダイアログが表示され…

文字を入力して[決定]ボタンをクリックすると…

JavaScript Sample - Windows Internet Explorer

C:¥JavaScript-Sa

Live Search

JavaScript Sample

Welcome!

鈴木一郎さん、ようこそ

コンピュータ | 保護モード: 無効 100%

入力された文字が親ページ上に表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; 
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css" 
    media="all">
```

```
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>Welcome!</h1>
    <div id="message"></div>  ―1
  </body>
</html>
```

1 …モーダルダイアログで入力された文字列を表示する領域を指定する

SOURCE CODE HTML name.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <script type="text/javascript" src="sub.js"></script>
  </head>
  <body>
    <h1>名前を入れてください</h1>
    <form>
      <input type="text" name="userName" id="userName">  ―1
      <input type="button" id="setButton" value="決定">
    </form>
  </body>
</html>
```

1 …名前を入力するテキストフィールドを指定する

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  var uName = showModalDialog("name.html",window,
  "dialogWidth:320px;dialogHeight:240px");              ―1
  document.getElementById("message").innerHTML =
  uName+"さん、ようこそ";                                 ―2
}
```

1 …モーダルダイアログを表示する

2 …入力された結果をページ上に表示する

SOURCE CODE JavaScript sub.js

```
window.onload = function(){
  document.getElementById("setButton").onclick = function(){
    window.returnValue = document.getElementById("userName").value; ─1
    window.close(); ─2
  }
}
```

1 …呼び出し元に必要なデータを渡す

2 …モーダルダイアログを閉じる

OnePoint モーダルダイアログを表示するには「showModalDialog()」を使う

モーダルダイアログの表示は、「showModalDialog()」を使います。ただし、Inernert Explorerでしか動作しないため注意が必要です。他のブラウザで同様の処理を行うには、Prototype Window Classなどのライブラリを利用します。

「showModalDialog()」の最初のパラメータに表示するHTML文書のURLを指定します。2番目のパラメータはwindow、3番目のパラメータは表示サイズなどのオプションを指定します。指定できるオプションは下表のようになります。

パラメータ名	初期値	内　容
dialogWidth	—	表示するウィンドウの横幅(ピクセル数を指定する)
dialogHeight	—	表示するウィンドウの縦幅(ピクセル数を指定する)
dialogLeft	—	表示位置(X座標。ピクセル数を指定する)
dialogTop	—	表示位置(Y座標。ピクセル数を指定する)
center	yes	画面中央に表示する/しない(表示する場合は「yes」または「1」または「on」、表示しない場合は「no」または「0」または「off」を指定する)
dialogHide	no	印刷/印刷プレビュー時に表示する/しない(表示する場合は「yes」または「1」または「on」、表示しない場合は「no」または「0」または「off」を指定する)
edge	raised	ウィンドウ枠形状(「sunken」は低い、「raised」は高い)
help	yes	Helpアイコンを表示する/しない(表示する場合は「yes」または「1」または「on」、表示しない場合は「no」または「0」または「off」を指定する)
resizable	no	リサイズ可能/不可能(可能にする場合は「yes」または「1」または「on」、可能にしない場合は「no」または「0」または「off」を指定する)
scroll	yes	スクロールする/しない(スクロールする場合は「yes」または「1」または「on」、スクロールしない場合は「no」または「0」または「off」を指定する)
status	no	ステータスバーを表示する/しない(表示する場合は「yes」または「1」または「on」、表示しない場合は「no」または「0」または「off」を指定する)
unadorned	no	外観指定(指定する場合は「yes」または「1」または「on」、指定しない場合は「no」または「0」または「off」を指定する)

「showModalDialog()」でデータが入力された内容を呼び出し元に渡すには、「window.returnValue」プロパティにデータを設定します。また、モーダルダイアログを閉じるには、「window.close()」と明示的に行う必要があります。

Internet Explorerでは「showModalDialog()」以外に、「showModelessDialog()」も用意されています。「showModelessDialog()」は「showModalDialog()」と異なり、表示後も呼び出し元のウィンドウなどにコントロールを移すことができます。

069 サブウィンドウを開く

ここでは、親ウィンドウのページを表示したタイミングでサブウィンドウを自動的に開く方法について解説します。

JavaScript Sample - Windows Internet Explorer
C:¥JavaScript-Sa
Live Search
JavaScript Sample

街の定食屋さん

今月のオススメをサブウィンドウで紹介しています。

JavaScript Sample - Windows Internet Explorer

今月のオススメ

当店自慢の豚の角煮です。ぜひ、ご賞味ください。

ページが読み込まれると、サブウィンドウが表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>街の定食屋さん</h1>
    <p>今月のオススメをサブウィンドウで紹介しています。</p>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.open("sub.html", "sbwin1","width=480,height=320");—1
```

1 …サブウィンドウを開く(サブウィンドウは480×320ピクセルでsub.htmlファイルが表示される)

OnePoint サブウィンドウを表示するには「window.open()」を使う

サブウィンドウを開くには、「window.open()」を使います。ただし、ブラウザの環境設定によっては表示がブロックされ、動作しないことがある点には要注意です。書式は、次のようになります。

```
window.open("URL","ウィンドウ名","オプション設定","ヒストリ設定")
```

サブウィンドウを開く「open()」の最初のパラメータにはサブウィンドウに表示するページのURLを指定します。2番目のパラメータはサブウィンドウの名前(ID名)を指定します。複数のサブウィンドウを開く場合には、この名前を別々にする必要があります。サブウィンドウの名前は、英数字で指定します。また、Internet Explorer5以降では、「_blank」「_parent」「_self」「_top」「search」も指定できます。Windows XP(SP1まで)とIE6の組み合わせでは「_media」も指定できます。

3番目のパラメータには、サブウィンドウの表示方法を文字列で指定します。指定できるオプションは下表のようになります。4番目のパラメータはサブウィンドウを開いた親ウィンドウのヒストリ(履歴)を継承するかどうかを指定します。「true」を指定すると継承し、「false」を指定すると継承しなくなります。

現在では、サブウィンドウを開くのは好ましくない傾向にあります。本当にサブウィンドウを利用しなければいけないのかどうか検討してから使用するようにしてください。また、別の方法としてPrototype Window Classライブラリを利用してページ内にウィンドウを開くという方法もあります。

◉サブウィンドウの表示方法のパラメータ

オプション名	初期値	内　容
toolbar	yes	ツールバーの表示/非表示(表示する場合は「yes」または「1」、非表示にする場合は「no」または「0」を指定する)
location	yes	ロケーションバーの表示/非表示(表示する場合は「yes」または「1」、非表示にする場合は「no」または「0」を指定する)
directories	yes	ディレクトリバーの表示/非表示(表示する場合は「yes」または「1」、非表示にする場合は「no」または「0」を指定する)
status	yes	ステータスバーの表示/非表示(表示する場合は「yes」または「1」、非表示にする場合は「no」または「0」を指定する)
menubar	yes	メニューバーの表示/非表示(Macを除く。表示する場合は「yes」または「1」、非表示にする場合は「no」または「0」を指定する)
scrollbars	yes	スクロールバーの表示/非表示(表示する場合は「yes」または「1」、非表示にする場合は「no」または「0」を指定する)
resizable	yes	ウィンドウサイズの変更可/変更不可(変更可にする場合は「yes」または「1」、変更不可にする場合は「no」または「0」を指定する)
width	—	ウィンドウの内側の横幅(ピクセル数を指定する)
height	—	ウィンドウの内側の高さ(ピクセル数を指定する)

オプション名	初期値	内　容
innerWidth	—	ウィンドウの内側の横幅(ピクセル数を指定する)
innerHeight	—	ウィンドウの内側の高さ(ピクセル数を指定する)
outerWidth	—	ウィンドウの外側の横幅(ピクセル数を指定する)
outerHeight	—	ウィンドウの外側の高さ(ピクセル数を指定する)
screenX	—	デスクトップの左端からの座標(ピクセル数を指定する)
screenY	—	デスクトップの上端からの座標(ピクセル数を指定する)
titlebar	yes	タイトルバーの表示/非表示(Macを除く。表示する場合は「yes」または「1」、非表示にする場合は「no」または「0」を指定する)
channelmode	no	チャンネルモード(有効にする場合は「yes」または「1」、無効にする場合は「no」または「0」を指定する)
fullscreen	no	全画面(フルスクリーン)モード※1(有効にする場合は「yes」または「1」、無効にする場合は「no」または「0」を指定する)
left	—	デスクトップの左端からの座標(ピクセル数を指定する)
top	—	デスクトップの上端からの座標(ピクセル数を指定する)
favorites	—	お気に入りの表示※2(表示する場合は「yes」または「1」、非表示にする場合は「no」または「0」を指定する)

※1　Windows XP SP2では完全なフルスクリーンモードにはなりません。
※2　Mac版Internet Explorerのみ有効です。
※※　Windows版Internet ExplorerではHTA(アプリケーション)でないと期待通り動作しないパラメータもあります。

◉オプションの各ブラウザの対応

オプション名	IE5	IE5.5	IE6	IE7	Firefox	Safari	Opera9
toolbar	○	○	○	○	○	—	—
location	○	○	○	○	○	○	○
directories	○	○	○	○	○	—	—
status	○	○	○	○	○	○	—
menubar	○	○	○	○	—	—	—
scrollbars	○	○	○	○	○	○	○
resizable	○	○	○	○	○	○	—
width	○	○	○	○	○	○	○
height	○	○	○	○	○	○	○
innerWidth	—	—	—	—	○	○	—
innerHeight	—	—	—	—	○	○	—
outerWidth	—	—	—	—	○	○	—
outerHeight	—	—	—	—	○	○	—
screenX	—	—	—	—	○	○	—
screenY	—	—	—	—	○	○	—
titlebar	○	○	○	○	—	—	—
channelmode	○	○	○	○	—	—	—
fullscreen	○	○	○	○	—	—	—
left	○	○	○	○	○	○	○
top	○	○	○	○	○	○	○
favorites	○	—	—	—	—	—	—

SECTION 070 サブウィンドウから親ウィンドウのページを切り替える

ここでは、表示されているサブウィンドウのリンクをクリックしたときに、親ウィンドウのページを切り替える方法について解説します。

サブウィンドウで*が付いていないリンクをクリックすると…

リンク先のページが親ウィンドウに表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; 
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css" 
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <p>*が付いていないリンク先はこのウィンドウに表示されます。</p>
  </body>
</html>
```

SOURCE CODE HTML sub.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; 
    charset=utf-8">
    <title>JavaScript Sample</title>
    <script type="text/javascript" src="sub.js"></script>
  </head>
  <body>
    <h1>お気に入りリンク</h1>
    <ul id="myfLink">
      <li><a href="http://www.yahoo.co.jp/" 
      class="ctrlWindow">Yahoo!</a></li>
      <li><a href="http://www.google.co.jp/" 
      class="ctrlWindow">Google</a></li>
      <li><a href="http://www.openspc2.org/">OpenSpace*</a></li>
      <li><a href="http://www.goo.ne.jp/" class="ctrlWindow">goo</a>
      <li><a href="http://www.excite.co.jp/">Excite*</a></li>
    </ul>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.open("sub.html", "sbwin1","width=480,height=320");—1
```

1 …サブウィンドウを開く

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  var root = document.getElementById("myfLink");
  var aTag = root.getElementsByTagName("a");       ─ 1
  for (var i=0; i<aTag.length; i++){
    if (aTag[i].className.match(/ctrlWindow/)){ ─ 2
      aTag[i].onclick = function(){               ─ 3
        window.opener.location.href = this.href;
        return false;
      }
    }
  }
}
```

1 …<a>タグの情報を取得し、タグの数だけ繰り返す

2 …<a>タグのスタイルシートクラスを調べて「ctrlWindow」クラスが設定されている場合のみ親ウィンドウへの操作を行うようにする

3 …クリックされたら「window.opener」を使って呼び出し元の親ウィンドウを操作する

OnePoint サブウィンドウから親ウィンドウを操作するには「window.opener」を使う

サブウィンドウから呼び出し元の親ウィンドウを操作するには、「window.opener」を使います。たとえば、「window.opener.document.bgColor」とすると、親ウィンドウの背景色を参照/設定することができます。サンプルのように親ウィンドウに任意のページURLを設定するには、「window.opener.location.href」と指定します。

親ウィンドウが閉じられている場合は、Internet Explorer以外では「window.opener」は「null」を返します。Internet ExplorerではObjectを返してしまうため、「window.opener.closed」プロパティの値を調べて親ウィンドウが閉じられているかどうか調べます。親ウィンドウが閉じられている場合には「true」、閉じられていない場合は「false」となります。

Column ウイルスガードソフトに注意する

ウイルスガードソフト(ウイルス対策ソフト)やインターネットセキュリティ関連のプログラムがインストールされている場合、JavaScriptのコードがページ上に追加されてしまい、予想とは違った動作を及ぼすことがあります。

このように勝手にプログラムが追加されたりする可能性も含めてJavaScriptコードを制作するというのも考慮しておくとよいでしょう。

SECTION 071

親ウィンドウからサブウィンドウのページを切り替える

ここでは、親ウィンドウに表示されているリンクをクリックしたとき、サブウィンドウに、そのリンク先のページを表示する方法について解説します。

JavaScript Sample - Windows Internet Explorer

お気に入りリンク集

- Yahoo!
- Google
- OpenSpace*
- goo
- Excite*

http://192.168.1.22/ - JavaScript Sample - Windows Interne...

*の付いていないリンク先はこのウィンドウに表示されます。

*が付いていないリンクをクリックすると…

http://www.google.co.jp/ - Google - Windows Internet Explo...

リンク先のページがサブウィンドウに表示される

SOURCE CODE　HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>お気に入りリンク集</h1>
    <ul id="myfLink">
      <li><a href="http://www.yahoo.co.jp/"
      class="ctrlWindow">Yahoo!</a></li>
      <li><a href="http://www.google.co.jp/"
   1  class="ctrlWindow">Google</a></li>
      <li><a href="http://www.openspc2.org/">OpenSpace*</a></li>
      <li><a href="http://www.goo.ne.jp/" class="ctrlWindow">goo</a>
      <li><a href="http://www.excite.co.jp/">Excite*</a></li>
    </ul>
  </body>
</html>
```

1 …サブウィンドウに表示するリンクの<a>タグの「class」属性に「ctrlWindow」を設定する(サブウィンドウに表示しない場合は設定しない)

SOURCE CODE　HTML sub.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
  </head>
  <body>
    <p>*の付いていないリンク先はこのウィンドウに表示されます。</p>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
var subWin =
window.open("sub.html","sbwin1","width=480,height=320");  [1]
window.onload = function(){
  var root = document.getElementById("myfLink");
  var aTag = root.getElementsByTagName("a");  [2]
  for (var i=0; i<aTag.length; i++){
    if (aTag[i].className.match(/ctrlWindow/)){  [3]
      aTag[i].onclick = function(){  [4]
        subWin.location.href = this.href;
        return false;
      }
    }
  }
}
```

1 …サブウィンドウを開く

2 …<a>タグの情報を取得し、タグの数だけ繰り返す

3 …<a>タグのスタイルシートクラスを調べて「ctrlWindow」クラスが設定されている場合のみサブウィンドウへの操作を行うようにする

4 …クリックされたらサブウィンドウのページURLを書き換える

OnePoint サブウィンドウを操作するには「window.open()」の戻り値を使う

親ウィンドウからサブウィンドウを操作するには、「window.open()」の戻り値を変数に入れておきます。この戻り値がサブウィンドウを操作するための情報になります。「window.open()」の戻り値はサブウィンドウを示すことになるため、サンプルのように変数「subWin」に「window.open()」の戻り値を入れた場合、「subWin.location.href」とすることでサブウィンドウのURLを変更することができます。同様に「subWin.document.bgColor」とすると、サブウィンドウの背景色を参照/設定することができます。

ただし、サブウィンドウの内容が別ドメインのページなどに更新された場合は、セキュリティの関係で参照することができなくなります。

また、サブウィンドウが閉じられてしまった場合には、操作することができずにエラーになります。サブウィンドウが閉じられているかどうかは、「closed」プロパティで調べることができます。たとえば、変数「subWin」に「window.open()」の戻り値が入っている場合、「subWin.closed」として調べることができます。「closed」プロパティが「true」であれば閉じられて、「false」であれば閉じられていないことになります。

Column タブはJavaScriptから制御できない

Firefox、Safari、Internet Explorer7など多くのブラウザは、同じウィンドウ内に別ページを表示するタブ機能があります。タブをクリックすることにより、異なるページを表示できます。JavaScriptではサブウィンドウを制御することができますが、タブで開かれているページにアクセスすることはできません。これはセキュリティの都合ではなく、単純にアクセスする手段が用意されていないためです。

SECTION 072

ページ内にマルチウィンドウを表示する

ここでは、Prototype Window Classライブラリを利用して、ページ内にマルチウィンドウを表示する方法について解説します。

ボタンをクリックすると…

ページ内にウィンドウが2枚表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
```

```
    <title>JavaScript Sample</title>
    <link rel="stylesheet" href="default.css" ↩
    type="text/css" media="all">
    <link rel="stylesheet" type="text/css" ↩
    href="main.css" media="all">
    <script type="text/javascript" src="prototype.js"></script>
    <script type="text/javascript" src="effects.js"></script>
    <script type="text/javascript" src="window.js"></script>
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>メッセージボード</h1>
    <form action="./multi.cgi" method="get" name="mainForm">
      <input type="button" id="openButton" value="メッセージを表示する">
    </form>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("openButton").onclick = function(){
    multiWindow.open("win01", "メッセージ1", 80, 100, 320, 240);   ┐1
    multiWindow.open("win02", "メッセージ2", 160, 160, 320, 240);  ┘
  }
}
var multiWindow = {
  open : function(winID, wTitle, x, y, w, h){
    var win = new Window(winID, {                          ┐
      title: wTitle, className: "dialog",                  │2
      top:y, left:x, width:w, height:h, zIndex: 100,       │
      resizable: true, draggable:true                      ┘
    });
    win.setDestroyOnClose(); ─3
    win.show(); ─4
    win.setHTMLContent("メッセージはありません。"); ─5
  }
}
```

1 …Prototype Window Classライブラリを利用してウィンドウを開く

2 …表示するウィンドウの名前やサイズを設定する

3 …ウィンドウが閉じられたらウィンドウに関するオブジェクトを削除するように設定する

4 …ウィンドウを表示する

5 …ウィンドウのページ内に文字を表示する

OnePoint ページ内にマルチウィンドウを表示するにはPrototype Window Classライブラリを利用する

Prototype Window Classライブラリを利用すると、手軽にページ内にマルチウィンドウを表示することができます。Prototype Window Classライブラリを動作させるにはprototype.jsライブラリとscript.aculo.usライブラリのエフェクト(effects.js)ファイルが必要なので、あらかじめ、prototype.jsライブラリとscript.aculo.usライブラリのファイルを読み込ませておきます。

ウィンドウを生成するには、「new Window()」とします。最初のパラメータにウィンドウIDを指定します。複数表示する場合には、このウィンドウIDが重複しないようにしてください。2番目のパラメータはオプションパラメータでプロパティリスト(ハッシュ)形式で指定します。指定できるパラメータは、下表のようになります。

また、ウィンドウが閉じられたら、ウィンドウ情報(ウィンドウオブジェクト)も破棄するように「setDestroyOnClose()」を指定します。これを指定しないと、内部ではウィンドウ情報が残ったままになります。ウィンドウを表示するには、「show()」を使います。表示されたウィンドウ内に文字などを表示するには、「setHTMLContent()」を使います。

ウィンドウのドラッグ、リサイズ、クローズなどの処理は自動的に行われるため、特に指定する必要はありません。

名　前	初期値	内　容
className	dialog	クラス名
title	none	ウィンドウタイトル
url	none	URL(別ページを表示する場合に指定)
parent	body	ウィンドウの親ノード
top	0	ウィンドウの表示Y座標(上)
bottom	—	ウィンドウの表示Y座標(下)
left	0	ウィンドウの表示X座標(左)
right	—	ウィンドウの表示X座標(右)
width	100	横幅
height	100	縦幅
maxWidth	none	最大幅(横)
maxHeight	none	最大幅(縦)
minWidth	100	最小幅(横)
minHeight	20	最小幅(縦)
resizable	true	リサイズ可能フラグ(true：リサイズ可能、false：リサイズ不可)
closable	true	クローズ可能フラグ(true：クローズ可能、false：クローズ不可)
minimizable	true	最小化ボタン(true：ボタンあり、false：ボタンなし)
maximizable	true	最大化ボタン(true：ボタンあり、false：ボタンなし)
draggable	true	ドラッグ可能フラグ(true：ドラッグ可能、false：ドラッグ不可)
showEffect	Effect.AppearまたはElement.show	ウィンドウ表示時のエフェクト
hideEffect	Effect.FadeまたはElement.hide	ウィンドウクローズ時のエフェクト
showEffectOptions	none	ウィンドウ表示時のエフェクトオプション(script.aculo.us参照)
hideEffectOptions	none	ウィンドウクローズ時のエフェクトオプション(script.aculo.us参照)
effectOptions	none	エフェクトオプション(script.aculo.us参照)
onload	none	読み込みが完了した時に呼び出す処理
opacity	1	不透明度(0〜1.0)

Prototype Window Classライブラリを利用してアラートダイアログを表示する

ここでは、Prototype Window Classライブラリを利用して、ページが読み込まれたタイミングで、ページ内にアラートダイアログを自動表示する方法について解説します。

JavaScript Sample - Windows Internet Explorer
C&R工房
ようこそ!
部屋を明るくしてから御覧ください
はい
コンピュータ | 保護モード: 無効 100%

ページが表示されるとアラートダイアログが表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; ⤵
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" href="default.css" ⤵
    type="text/css" media="all">
    <link rel="stylesheet" href="alert.css" type="text/css" ⤵
    media="all">
    <link rel="stylesheet" type="text/css" href="main.css" ⤵
    media="all">
    <script type="text/javascript" src="prototype.js"></script>
    <script type="text/javascript" src="effects.js"></script>
    <script type="text/javascript" src="window.js"></script>
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
```

1

```
    <h1>C&amp;R工房</h1>
    ようこそ！
  </body>
</html>
```

1 …必要なライブラリとスタイルシートファイルを読み込む

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  Dialog.alert("部屋を明るくしてから御覧ください", {
    windowParameters: {width:320, height:100},
    okLabel: "はい"
  });
}
```

1 …アラートダイアログを表示する

OnePoint アラートダイアログを表示するにはPrototype Window Classライブラリの「Dialog.alert()」を使う

Prototype Window Classライブラリには、アラートダイアログを表示するための「Dialog.alert()」メソッドが用意されています。最初のパラメータに表示するメッセージを指定します。2番目のパラメータにオプションを指定します。指定できるオプションは、下表のようになります。[OK]ボタンがクリックされたときの処理などを指定することができます。

オプション	内　容
top	Y座標
left	X座標
okLabel	OKボタンに表示する文字列
ok	OKボタンがクリックされたときの処理
buttonClass	OKボタンに適用するCSSクラス名

Column JavaScriptのグラフィックスライブラリ

JavaScriptでは、グラフィックス処理を行うことができません。HTMLタグの<canvas>と組み合わせることでグラフィックスを扱うことができますが、Internet Explorerでは<canvas>タグがサポートされていません。多くのブラウザで長方形や楕円など基本図形を描画したい場合には、次のグラフィックスライブラリが利用できます。

URL http://www.walterzorn.com/index.htm

074 Prototype Window Classライブラリを利用して確認ダイアログを表示する

ここでは、Prototype Window Classライブラリを利用して、ページが読み込まれたタイミングで、ページ内に確認ダイアログを自動表示する方法について解説します。

JavaScript Sample - Windows Internet Explorer

映画批評

ようこそ。ネタバレに注意してください。

本当にこのページを閲覧しますか?

はい　いいえ

ページが表示されると確認ダイアログが表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" href="default.css"
    type="text/css" media="all">
    <link rel="stylesheet" href="alert.css"
    type="text/css" media="all">
    <link rel="stylesheet" type="text/css"
    href="main.css" media="all">
    <script type="text/javascript" src="prototype.js"></script>
    <script type="text/javascript" src="effects.js"></script>
    <script type="text/javascript" src="window.js"></script>
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
```

```
    <h1>映画批評</h1>
    ようこそ。ネタバレに注意してください。
  </body>
</html>
```

1 …必要なライブラリとスタイルシートファイルを読み込む

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  Dialog.confirm("本当にこのページを閲覧しますか?", {
    windowParameters: {width:300, height:60},
    okLabel: "はい",
    cancelLabel: "いいえ",
    ok:function(win){ alert("では御覧ください"); return true; },
    cancel:function(win){ alert("閲覧を中止します"); return true; }
  });
}
```

1 …確認ダイアログを表示する

OnePoint 確認ダイアログを表示するにはPrototype Window Classライブラリの「Dialog.confirm()」を使う

Prototype Window Classライブラリには、確認ダイアログを表示するための「Dialog.confirm()」メソッドが用意されています。最初のパラメータには、表示するメッセージを指定します。2番目のパラメータには、オプションを指定します。指定できるオプションは下表のようになります。[OK]ボタンや[キャンセル]ボタンがクリックされたときの処理などを指定することができます。

オプション	内　容
top	Y座標
left	X座標
okLabel	OKボタンに表示する文字列
cancelLabel	キャンセルボタンに表示する文字列
ok	OKボタンがクリックされたときの処理
cancel	キャンセルボタンがクリックされたときの処理
buttonClass	OKボタンに適用するCSSクラス名

Column 情報ウィンドウ

Prototype Window Classは情報ウィンドウを表示することができます。これは次のように「Dialog.info()」を使います。

```
Dialog.info("現在ダウンロード中です", {
windowParameters: {width:300, height:60},
showProgress: true
});
```

SECTION 075

月別にサブウィンドウにファイルや画像を表示する

ここでは、月ごとに異なるファイルや画像を、サブウィンドウに表示する方法について解説します。

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>徒然日記</h1>
    <p>今月の一言は別ウィンドウで表示しています。</p>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
var dateObj = new Date();—1
var n = dateObj.getMonth() + 1;—2
window.open("pages/"+n+".html", "sbwin","width=480,height=320");—3
```

1 …「Date」オブジェクトを生成する
2 …現時点での月を求める(「getMonth()」は1少ない値を返すので1を加算する)
3 …求めた現時点での月の値とファイル名を組み合わせたパスを指定する

OnePoint ファイル名と月の値を組み合わせる

月別にサブウィンドウに表示するページを変えるには、あらかじめ月別にファイルを用意しておきます。この時、ファイル名に工夫をし、月の値を含めるようにします。たとえば、ファイル名を「1.html」～「12.html」のようにしておきます。サブウィンドウを開く際に「Date」オブジェクトで現在の月を求めます。ファイル名と求めた月の値を組み合わせたパスをサブウィンドウの最初のパラメータに指定することで月別に異なるページをサブウィンドウに表示することができます。

Break Time

関数定義の動作の違い

「function」を使って関数を定義する場合に、すでに関数があるかどうか調べてから処理したい場合があります(関数のオーバーライド)。通常は「if」命令または「||」で判定処理を行ってから定義・再定義します。しかし、次のサンプルのような場合、Internet Explorerと他ブラウザでは定義される関数が異なります。

```
var testFunc;
function testFunc2(){
  alert("最初に定義した関数");
}
alert(testFunc);
if (!testFunc) {
  function testFunc(){
    alert("二番目に定義した関数");
  }
}else{
  function testFunc(){
    alert("三番目に定義した関数");
  }
}
testFunc();
```

Internet Explorerではスクリプトの処理に関係なく、最後に「function」で記述された関数が定義されます。他のブラウザでは実際のスクリプトの処理に従って定義されます。上記のサンプルを実行するとInternet Explorerでは、次の画面のようにアラートダイアログに関数の内容が表示された後に「三番目に定義した関数」と表示されます。他のブラウザでは「undefined」と表示された後に「二番目に定義した関数」と表示されます。このようなブラウザによる関数定義の違いには、注意してください。

CHAPTER - 06

DOM

特定のIDを持つオブジェクトにアクセスする

ここでは、ページ上の特定のIDを持つオブジェクトにアクセスする方法について解説します。

入力された文字がページ上の領域に表示される

テキストフィールドに文字を入力してボタンをクリックすると…

SOURCE CODE　HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>伝言板</h1>
    <form action="./write.cgi" method="get" name="mainForm">
      書き込む内容:<input type="text" name="writeData"
      id="writeData" value="">                              1
      <input type="button" id="setButton" value="書き込む">
    </form>
```

```
    <div id="message"></div>
  </body>
</html>
```

1 …読み出すテキストフィールドには「name」属性と「id」属性を指定する
2 …入力された内容を表示する<div>タグにもID名を指定する

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("setButton").onclick = function(){
    var text = document.getElementById("writeData").value; —1
    document.getElementById("message").innerHTML = text; —2
  }
}
```

1 …ID名が「writeData」のエレメント(「id」属性が「writeData」のテキストフィールド)の内容を読み出す
2 …ID名が「message」のエレメント(「id」属性が「messeage」の<div>タグ)にテキストフィールドの内容を書き出す

OnePoint ID名の取得は「getElementById()」を使う

特定のID名を持つページ上のエレメントにアクセスするには、「document.getElementById()」を使います。パラメータには、取得したいエレメントのID名を指定します。「document.getElementById()」の戻り値はエレメントへの参照(ポインタ)になります。後はエレメントごとのメソッドやプロパティを参照/設定することで各種処理を行うことができます。

Column テーブルのノード(tbody)に注意

テーブルタグはHTML文書上では定義されていなくても、内部では正しいタグ付けに変換されます。たとえば、次のサンプルのHTML文書には<tbody>タグは記述されていませんが、内部では<tbody>タグが追加された文書構造として定義されます。HTML文書で正しく定義、省略しなければよいのですが、これを忘れてしまうと異なるノードを参照してしまい、間違いのもとになってしまいます。なお、ノードなどの確認はDOMツリー表示が可能なブラウザであれば原因判明の糸口になるでしょう。

◉HTMLファイル

```
<table id="table1" border="1"><tr><th>CPU</th></tr>
<tr><td>Xeon</th></tr></table>
```

◉JavaScriptファイル

```
window.onload = function(){
  tbl = document.getElementById("table1");
  alert(tbl.firstChild.tagName);
}
```

SECTION

077 特定のタグ名を持つオブジェクトを制御する

ここでは、ページ上の「li」のタグ名を持つオブジェクト（<li>タグ）を制御する方法について解説します。

リストの文字上にマウスカーソルを重ねると、背景色が変化する（マウスカーソルを離すと元に戻る）

SOURCE CODE　HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
```

```
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>用語解説</h1>
    <ul>
      <li>オブジェクトとは?</li>
      <li>メソッドとは?</li>
      <li>プロパティとは?</li>
      <li>イベントとは?</li>
    </ul>
    <ul>
      <li>CSSとは?</li>
      <li>XSLTとは?</li>
    </ul>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  var liTag = document.getElementsByTagName("li");
  for (var i=0; i<liTag.length; i++){
    liTag[i].onmouseover = function(){
      this.style.backgroundColor = "#faa";
    }
    liTag[i].onmouseout = function(){
      this.style.backgroundColor = "#fff";
    }
  }
}
```

1 …<li>タグだけ抽出し、その数だけ繰り返す

2 …<li>タグにマウスが重なったときに背景色を薄赤にするようにする

3 …<li>タグからマウスが離れたときに背景色を白色(元に戻す)にするようにする

OnePoint 「getElementsByTagName()」を使ってタグ情報を取得する

特定のタグを取得するには「getElementsByTagName()」を使い、パラメータには取得したいタグ名を指定します。結果は配列として返され、それぞれHTML文書内での出現順に格納されます。配列に取得したタグ情報が入っているので、マウスイベントなどを設定し、各種処理を行わせることができます。サンプルでは、マウスオーバー時とマウスアウト時に<li>タグの背景色を変えるようにしています。サンプルでの「this」は<li>タグ自身を示すため、「this.style.backgroundColor」とすると、マウスイベントが発生した<li>タグを示すことになります。

078 特定の名前を持つオブジェクトを制御する

ここでは、ページ上で「osType」という名前を持つオブジェクトの情報を取得する方法について解説します。

JavaScript Sample - Windows Internet Explorer

利用しているOSの調査

MacOS X
Linux
Windows
選択

結果：

ラジオボタンをONにして[選択]ボタンをクリックすると…

選択された項目が表示される

Windowsが選択されています。

SOURCE CODE　HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
```

```
  <body>
    <h1>利用しているOSの調査</h1>
    <form action="./check.cgi" method="get" 
    id="mainForm" name="mainForm">
      <input type="radio" name="osType" value="MacOS X" 
      checked>MacOS X<br>
      <input type="radio" name="osType" value="Linux">Linux<br>
      <input type="radio" name="osType" value="Windows">Windows<br>
      <input type="button" id="checkButton" value="選択">
    </form>
    <div id="result">結果:</div>
  </body>
</html>
```

1 …ラジオボタンには「name」属性を指定して同じ名前(グループ名)にする

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("checkButton").onclick = function(){
    var radio = document.getElementsByName("osType"); —1
    for (var i=0; i<radio.length; i++){ —2
      if (radio[i].checked) {
        document.getElementById("result").innerHTML = 
        radio[i].value+"が選択されています。";
      }
    }
  }
}
```

1 …「name」属性が「osType」のタグを抽出する

2 …ラジオボタンの数だけ繰り返す

3 …ラジオボタンがチェックされているかどうか「chekced」プロパティを調べ、チェックされていたらページ上に選択項目を表示する

OnePoint 「document.getElementsByName()」を使ってタグ情報を取得する

「name」属性で指定された名前を取得するには、「document.getElementsByName()」を使います。これは主に入力フォームのラジオボタンへアクセスするために利用します。ラジオボタンは同じ「name」属性を持つグループとして定義されます。「document.getElementsByName()」を使えば、指定された名前を持つタグの情報を配列として返します。

サンプルでは「osType」という名前を持つタグ(ラジオボタン)を取得し、選択されているかどうかを「checked」プロパティを使って調べています。「checked」プロパティが「true」であればON、「false」であればOFFになっていることになります。

SECTION

079 特定のクラス名を持つオブジェクトを制御する

ここでは、prototype.jsライブラリを使って、「note」というクラス名を持つオブジェクトの枠線を変更する方法について解説します。

JavaScript Sample - Windows Internet Explorer

C:¥JavaScript-Sa

Live Search

JavaScript Sample

ボタンをクリックすると、noteクラスだけ枠が太くなります。

noteクラスだけ枠を太くする

ここは noteクラス

ここは cautionクラス

ここは noteとcautionクラス

ここもnoteクラス

ここは cautionクラス

コンピュータ | 保護モード: 無効

ボタンをクリックすると…

スタイルシートクラスが「note」のボックスだけ、枠が太くなる

JavaScript Sample - Windows Internet Explorer

C:¥JavaScript-Sa

Live Search

JavaScript Sample

ボタンをクリックすると、noteクラスだけ枠が太くなります。

noteクラスだけ枠を太くする

ここは noteクラス

ここは cautionクラス

ここは noteとcautionクラス

ここもnoteクラス

ここは cautionクラス

コンピュータ | 保護モード: 無効 100%

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
```

```
    <script type="text/javascript" src="prototype.js"></script>
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    ボタンをクリックすると、noteクラスだけ枠が太くなります。
    <form action="./access.cgi" method="get"
    id="mainForm" name="mainForm">
      <input type="button" id="setButton" value="noteクラスだけ枠を太くする">
    </form>
    <div class="note">ここはnoteクラス</div>
    <div class="caution">ここはcautionクラス</div>
    <div class="note caution">ここはnoteとcautionクラス</div>
    <div class="note">ここもnoteクラス</div>
    <div class="caution">ここはcautionクラス</div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("setButton").onclick = function(){
    $$(".note").each(function(obj){ —1
      obj.style.borderWidth = "4px"; —2
    });
  }
}
```

1 …「$$()」を使ってスタイルシートクラス名が「note」のエレメントだけを取得し、処理を行う

2 …スタイルシートの枠の幅を4ピクセルに設定する

OnePoint 特定のクラス名の取得には「$$()」を使う

prototype.js ver 1.5以降では、特定のスタイルシートクラス名を持つエレメントだけを取得することができます。たとえば、「$$(".note")とすると、スタイルシートの「note」クラスが適用されているエレメントだけを取得することができます。取得したエレメントに対して処理を行いますが、prototype.jsでは「for()」ではなく、「each()」を使って繰り返し処理を行うことができます(イテレータ)。「each()」を使うことで繰り返しに必要は変数や繰り返し回数などを記述する必要がなくなり、記述ミスによる不具合が減少します。

「each()」で繰り返す際に呼び出す関数には、「$$()」で取得したエレメントへの参照がパラメータとして渡されます。サンプルでは渡されたエレメントのスタイルシートを操作することで枠の太さを変更しています。

SECTION 080

特定の階層構造を持つオブジェクトを制御する

ここでは、prototype.jsライブラリを使って、<div>タグと<span>タグの階層構造を持つオブジェクトの文字色を変更する方法について解説します。

ボタンをクリックすると、divとspanの階層だけ文字が赤くなります。
div span階層だけ文字色を赤色にする
ここは divのみ
divとspanの階層
divとspanの階層ここは div
ここは divでここは divとspanの階層
ここはpとspan

ボタンをクリックすると、<div><span>タグの階層になっている部分の文字色が赤色になる

SOURCE CODE HTML ▶ index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; ↩
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css" ↩
    media="all">
    <script type="text/javascript" src="prototype.js"></script>
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    ボタンをクリックすると、divとspanの階層だけ文字が赤くなります。
    <form action="./access.cgi" method="get" ↩
    id="mainForm" name="mainForm">
```

```
    <input type="button" id="setButton"
    value="div span階層だけ文字色を赤色にする">
  </form>
  <div id="contents">
    <div class="note">ここはdivのみ</div>
    <div class="note"><span>divとspanの階層</span></div>
    <div class="note"><span>divとspanの階層</span>ここはdiv</div>
    <div class="note">ここはdivで<span>ここはdivとspanの階層</span></div>
    <p class="note"><span>ここはpとspan</span></p>
  </div>
 </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("setButton").onclick = function(){
    $("contents").getElementsBySelector
    ("div span").each(function(obj){    ― 1
      obj.style.color = "red"; ― 2
    });
  }
}
```

1 …「getElementsBySelector()」を使って階層構造を指定し、該当するセレクタだけを「each()」で繰り返す

2 …スタイルシートの「color」プロパティに「red」を指定する(文字色が赤になる)

OnePoint 特定の階層構造の取得には「getElementsBySelector()」を使う

prototype.js ver 1.5以降では、特定の階層構造を持つエレメントだけを取得することができます。「getElementsBySelector("div span")」とすると、子孫セレクタの指定になります。このセレクタの指定は、スタイルシートの定義の際に指定できるセレクタが利用できます。

「getElementsBySelector("div span")」で取得したエレメントに対して処理を行いますが、prototype.jsでは「for()」ではなく、「each()」を使って繰り返し処理を行うことができます(イテレータ)。「each()」を使うことで繰り返しに必要は変数や繰り返し回数などを記述する必要がなくなり、記述ミスによる不具合が減少します。

「each()」で繰り返す際に呼び出す関数には、「getElementsBySelector()」で取得したエレメントへの参照がパラメータとして渡されます。サンプルでは、渡されたエレメントのスタイルシートを操作することで文字を赤色に変更しています。

Column jQueryでも特定の階層構造を取得することができる

jQueryでも階層構造を指定してエレメントを取得することができます。jQueryの場合、prototype.jsと異なり、「$()」を使います。「$("div span")」のように指定します。jQueryではCSS Level 3で指定することができるセレクタにも対応しています。

081 ページにテキストを出力する

ここでは、テキストフィールドに入力した文字を、ページ上にテキストとして出力する方法について解説します。

JavaScript Sample - Windows Internet Explorer
メッセージボード
書き込む内容: <p>タグについて
書き込む

テキストフィールドに文字を入力してボタンをクリックすると…

JavaScript Sample - Windows Internet Explorer
メッセージボード
書き込む内容: <p>タグについて
書き込む
<p>タグについて

入力された内容がそのままページ上に表示される

SOURCE CODE　HTML ▶ index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
```

```
  <body>
    <h1>メッセージボード</h1>
    <form action="./write.cgi" method="get" name="mainForm">
      書き込む内容:<input type="text" name="writeData"
      id="writeData" value="">
      <input type="button" id="setButton" value="書き込む">
    </form>
    <div id="message"></div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("setButton").onclick = function(){
    var text = document.getElementById("writeData").value; —1
    document.getElementById("message").innerText = text; —2
    document.getElementById("message").textContent = text; —3
  }
}
```

1 …テキストフィールドの内容を読み出す
2 …Firefox以外のブラウザではinnerTextプロパティに表示する文字列を指定する
3 …FirefoxではtextContentプロパティに表示する文字列を指定する

OnePoint ページ上にプレーンなテキストを表示するには「textContent」または「innerText」を使う

ページ上にプレーンなテキストを表示するには、「textContent」プロパティまたは「innerText」プロパティに書き込みます。Firefoxでは「innerText」プロパティがないため、「textContent」プロパティに書き込み必要があります。Firefox以外のブラウザでは「innerText」プロパティに表示したい文字列を指定します。プレーンなテキストとして処理されるため、文字列内にHTMLタグがある場合でもタグは反映されず、そのままの文字が表示されます。

Column 多言語の表示

HTML文書では、英語や日本語以外に多くの言語に対応できるようになっています。UTF-8であれば、他の国の文字も混在して文章に表示させることができます。JavaScriptからでも次のようにしてページ内に他の国の文字を表示させることができます。なお、下記はハングル、中国語、アラビア文字の3文字をページ上に表示する記述です。

```
var text = "&#12593;"+"&#63906;"+"&#64336;";
document.getElementById("message").innerHTML = text;
```

082 ページにHTMLデータを出力する

ここでは、テキストフィールドに入力したHTMLデータをページ上に反映する方法について解説します。

JavaScript Sample - Windows Internet Explorer

HTMLタグの確認

書き込む内容: 赤は

書き込む

テキストフィールドに文字を入力してボタンをクリックすると…

HTMLタグが反映された状態で入力された内容がページ上に表示される

赤は

SOURCE CODE　HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
```

```
    <script type="text/javascript" src="main.js"></script>
  </head>

  <body>
    <h1>HTMLタグの確認</h1>
    <form action="./write.cgi" method="get" name="mainForm">
      書き込む内容:<input type="text" name="writeData"
      id="writeData" value="">
      <input type="button" id="setButton" value="書き込む">
    </form>
    <div id="message"></div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("setButton").onclick = function(){
    var text = document.getElementById("writeData").value; —1
    document.getElementById("message").innerHTML = text; —2
  }
}
```

1 …テキストフィールドの内容を読み出す
2 …「innerHTML」プロパティに表示する文字列を指定する

OnePoint HTMLタグを反映させて表示したい場合は「innerHTML」プロパティを使って表示する

ページ上にHTMLタグも反映させて表示させたい場合には、「innerHTML」プロパティに表示する文字およびHTMLタグを書き込みます。「innerHTML」プロパティにHTMLタグを含むデータを書き込むと、ほとんどのタグは正しく反映されます。ただし、入力フォームなど一部のタグに関しては設定した属性などが反映されないことがあります。また、書き込むタグが閉じられていない場合には強制的に閉じタグが付加されます。

「innerHTML」プロパティへの書き込み処理は低速なため、変数などに入れておき最後に一括して書き込むようにします。

Column Internet Explorerでの文字列の連結について

Internet Explorerでは「innerHTML」プロパティに書き込む文字列を「+」記号で連結するよりも、連結する文字列を配列に格納し、最後に「join()」で連結した後に書き込む方が高速に処理されます(282ページ参照)。

083 ノードの値を参照/設定する

ここでは、ノードの値を参照/設定する方法について解説します。

ボタンをクリックすると…

ノードの値が参照/設定される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
```

```
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    ボタンをクリックすると、ノードの値を参照/設定します
    <form action="./checknode.cgi" method="get" name="mainForm">
      <input type="text" name="fNode" id="fNode"
      value="farming route">
      <input type="button" id="setButton" value="各ノードの値を参照/設定">
    </form>
    <div id="dNode">ここはテキストノードです</div>   ―1
    <div id="result"></div>
  </body>
</html>
```

1 …操作するノードにID名(fNode)を指定する

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("setButton").onclick = function(){
    var text = document.getElementById("fNode").value;
    document.getElementById("result").innerHTML += text +"<br>";
    text = document.getElementById("dNode").firstChild.nodeValue;   ―1
    document.getElementById("result").innerHTML += text +"<br>";
    document.getElementById("fNode").value = "農道";
    document.getElementById("dNode").firstChild.nodeValue =   ┐
    "書き換えました";                                           ┘2
  }
}
```

1 …ID名が「dNode」の最初のノード値を読み出す(表示されている文字列が読み出される)

2 …ID名が「dNode」の最初のノード値を書き換える

OnePoint ノードの値は「nodeValue」で参照や設定を行う

ノードの値は、「nodeValue」を使って参照/設定することができます。ノードの指定は、基準となる位置をID名で指定しておくと便利です。ドキュメントルートからたどるよりも、操作対象となるノードまたは親ノードにID名を指定しておくと楽になります。また、遠くのノードからたどるよりも近くのノードからたどる方がブラウザによるノード処理の違いに悩まされなくて済みます。

ノードの位置指定は、「childNodes[]」「firstChild」「lastChild」などを利用します。タグも表示されているテキストや改行もノードになりますが、Internet Explorerではタグ間にある空白やタブ、改行はテキストノードとして処理されず存在しないことになります。これに対してFirefoxやSafari、Operaなどではテキストノードとして処理されるため、ノードを単純にたどっていくと異なるノードを参照してしまうことがあるので注意してください。

SECTION

084 ノードの種類を読み出す

ここでは、ノードの種類を読み出す方法について解説します。

JavaScript Sample - Windows Internet Explorer

ノードの種類を数値で表示します

ノードの種類を表示

ここはテキストノードです

ボタンをクリックすると…

一連のノードの種類が表示される

3
8
1

SOURCE CODE　HTML ▶ index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; 
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css" 
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
```

```
  <body>
    ノードの種類を数値で表示します
    <form action="./checknode.cgi" method="get" name="mainForm">
      <input type="button" id="setButton" value="ノードの種類を表示">
    </form>
    <div id="dNode">ここは<!-- コメント --><span>テキストノード</span>です</div>
    <div id="result"></div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("setButton").onclick = function(){
    var d = document.getElementById("dNode"); —1
    var res = document.getElementById("result");
    for (var i=0; i<3; i++){ —2
      var nType = d.childNodes[i].nodeType; —3
      res.innerHTML += nType +"<br>";
    }
  }
}
```

1 …基準となるノード(ID名「dNode」)を取得する

2 …3つのノード分繰り返す

3 …ノードの種類を取得する

OnePoint ノードの種類は「nodeType」で取得する

ノードの種類は「nodeType」を使って取得することができます。「nodeType」は、下表の値を返します。ただし、Safari2ではコメントノードは取得することができません。

nodeType値	名　称	内　容
1	ELEMENT_NODE	エレメントノード
2	ATTRIBUTE_NODE	属性ノード
3	TEXT_NODE	テキストノード
4	CDATA_SECTION_NODE	CDATAセクションノード
5	ENTITY_REFERENCE_NODE	エンティティリファレンス
6	ENTITY_NODE	エンティティ
7	PROCESSING_INSTRUCTION_NODE	処理命令
8	COMMENT_NODE	コメントノード
9	DOCUMENT_NODE	ドキュメントノード
10	DOCUMENT_TYPE_NODE	ドキュメントタイプ
11	DOCUMENT_FRAGMENT_NODE	ドキュメントフラグメント
12	NOTATION_NODE	記法

085 ノードのタグ名を読み出す

ここでは、ノードのタグ名を取得する方法について解説します。

JavaScript Sample - Windows Internet Explorer
C:¥JavaScript-Sa
Live Search
JavaScript Sample
ボタンをクリックすると、ノードのタグ名を表示します
ノードのタグ名を表示
ここはテキストノードです
コンピュータ | 保護モード: 無効

JavaScript Sample - Windows Internet Explorer
C:¥JavaScript-Sa
Live Search
JavaScript Sample
ボタンをクリックすると、ノードのタグ名を表示します
ノードのタグ名を表示
ここはテキストノードです
undefined
!
SPAN
コンピュータ | 保護モード: 無効
100%

ボタンをクリックすると、一連のノードのタグ名が表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
```

```
  <body>
    ボタンをクリックすると、ノードのタグ名を表示します
    <form action="./checknode.cgi" method="get" name="mainForm">
      <input type="button" id="setButton" value="ノードのタグ名を表示">
    </form>
    <div id="dNode">ここは<!-- コメント --><span>テキストノード</span>です</div>
    <div id="result"></div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("setButton").onclick = function(){
    var d = document.getElementById("dNode"); —1
    var res = document.getElementById("result");
    for (var i=0; i<3; i++){ —2
      var nType = d.childNodes[i].tagName; —3
      res.innerHTML += nType +"<br>";
    }
  }
}
```

1 …基準となるノード(ID名「dNode」)を取得する

2 …3つのノード分繰り返す

3 …ノードのタグ名を取得する

OnePoint タグ名は「tagName」で取得する

ノードがタグの場合、「tagName」プロパティでタグ名を取得することができます。たとえば、テキストノードなど、ノードがタグでない場合には、「tagName」プロパティは「undefined」となります。サンプルを実行すると、ブラウザごとに結果が異なります(下図参照)。テキストノードの場合は同じですが、Safari2ではコメントが無視され、IE7では「!」を返してしまいます。このようなブラウザの違いには注意が必要です。

また、タグ名を取得する場合には、「nodeType」の値を取得しノードがタグかどうかを調べてから取得するようにしてください。

●Safari2の場合

●Firefox2の場合

SECTION

086 ノードの属性値を参照/設定する

ここでは、ノードの属性値を参照/設定する方法について解説します。

ボタンをクリックすると、リンクの「target」属性がfindに設定され…

リンクをクリックすると…

新しいウィンドウ(またはタブ)でリンク先が表示される

SOURCE CODE　HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
```

```
  <body>
    <h1>リンク集</h1>
    すべてのリンクを新しいウィンドウ(またはタブ)で表示したい場合はボタンをクリックしてください。<br>
    ※下の2つは、はじめから新しいウィンドウ(またはタブ)で表示されます。
    <form action="./checknode.cgi" method="get" name="mainForm">
      <input type="button" id="setButton" value="別窓で開く"><br>
    </form>
    <ul id="linkList">
      <li><a href="http://www.yahoo.co.jp/">Yahoo!</a></li>
      <li><a href="http://www.google.co.jp/">Google</a></li>
      <li><a href="http://www.openspc2.org/"
      target="find">OpenSpace</a></li>
      <li><a href="http://www.goo.ne.jp/" target="find">goo</a></li>
    </ul>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("setButton").onclick = function(){
    var linkTag = document.getElementById
    ("linkList").getElementsByTagName("a");   ―1
    for (var i=0; i<linkTag.length; i++){   ―2
      var tgt = linkTag[i].getAttribute("target");   ―3
      if (tgt != "find"){
        linkTag[i].setAttribute("target", "find");   ―4
      }
    }
  }
}
```

1 …ID名「linkList」内にある<a>タグの情報を取得する

2 …<a>タグの数だけ繰り返す

3 …「target」属性の値を読み出す

4 …「target」属性で「find」が設定されていない場合は、ターゲットを「find」に設定する

OnePoint 属性の読み出しは「getAttribute()」、設定は「setAttribute()」で行う

タグの属性を読み出すには、「getAttribute()」を使います。パラメータには取得したい属性名を指定します。属性値を設定したい場合には、「setAttribute()」を使います。最初のパラメータに設定する属性名、2番目のパラメータに設定する値を指定します。

サンプルではページが読み込まれると<a>タグの「target」属性が指定されているかどうか調べ、指定されていない場合は「target」属性を設定するようにします。これにより、リンクをクリックすると新たなウィンドウ(またはタブ)で内容が表示されます。

SECTION 087 ノードを複製して追加する

ここでは、ノードを複製して追加する方法について解説します。

ボタンをクリックすると…

複製元のノードが追加され、結果の領域にコピーされる

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
```

```
  <body>
    ボタンをクリックすると、ノードを複製して追加します。
    <form action="./dupnode.cgi" method="get" name="mainForm">
      <input type="button" id="dupButton" value="複製して追加する"><br>
    </form>
    <div id="dNode" class="dup"><span>複製元のテキストです</span></div>
    <div id="result">結果:</div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("dupButton").onclick = function(){
    var d = document.getElementById("dNode"); —1
    d = d.cloneNode(true); —2
    d.id = "id"+(new Date()).getTime(); —3
    document.getElementById("result").appendChild(d); —4
  }
}
```

1 …複製元の情報を読み出す(ID名「dNode」)

2 …複製元として読み出したノードを複製する(パラメータに「true」を指定して子ノードごと複製する)

3 …ページ内で重複しない新たなIDを割り当てる

4 …結果を表示する領域に追加する

OnePoint ノードを複製するには「cloneNode()」を使う

ノードを複製するには、「cloneNode()」を使います。パラメータに「true」を指定すると、ノードに含まれる子ノード以下すべてが複製の対象になります。「false」を指定すると、複製時に子ノードは含まれなくなります。複製したノードは、そのままではページ上には表示されません。ページ上に表示するには、ページ上にあるノードのいずれかに連結する必要があります。サンプルでは、「appendChild()」を使ってノードを追加しています。

なお、ノードを複製した後には、必ず重複しないID名を割り当ててからページ上に表示する(連結する)ようにしてください。ID名が重複すると、正しく動作しなくなります。

Column ノードの挿入

「appendChild」はノードを末尾に追加しますが、指定したノードの前にノードを挿入したい場合には「insertBefore()」を使います。また、prototype.jsを使えば、次のように手軽にノードを挿入することができます。他にもprototype.jsには「Insertion.After」「Insertion.Bottom」「Insertion.Top」が用意されています。

```
new Insertion.Before("result", "サンプル");
```

088 ノードを削除する

ここでは、ノードを削除する方法について解説します。

ボタンをクリックすると…

領域内の最初のノードが削除される

SOURCE CODE　HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
```

```
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    ボタンをクリックすると、ノードを削除します。
    <form action="./rmnode.cgi" method="get" name="mainForm">
      <input type="button" id="rmButton" value="ノードを削除"><br>
    </form>
    <div id="dNode" class="rm"><span>Node1</span>,<span>Node2</span>,
    <span>Node3</span>,<span>Node4</span>,<span>Node5</span>
    </div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("rmButton").onclick = function(){
    var d = document.getElementById("dNode"); —1
    d.removeChild(d.firstChild); —2
  }
}
```

1 …削除するノードを含む領域の情報を読み出す(ID名dNode)

2 …最初のノードを削除する

OnePoint ノードを削除するには「removeChild()」を使う

ノードを削除するには、「removeChild()」を使います。パラメータに削除するノードを指定します。サンプルでは、パラメータに「firstChild」を指定して最初のノードを削除しています。最後のノードを削除したいのであれば、パラメータに「lastChild」を指定します。

なお、削除するノードが存在しない場合は、エラーになります。これを防ぐには、「childNodes.length」として子ノードの数を調べてから削除するようにします。

Column 複数のノードを削除する

複数のノードを削除する場合には、「for()」を使って繰り返しノードの削除処理を行います。ただし、このときに次のようにしてしまうと、期待通りに動作しなかったりエラーになってしまうことがあります。

```
for (var i=0; i<ノードの長さ; i++) {
  削除処理
}
```

これは最初からノードをたどって削除すると、削除した時点でノードの参照位置が変わってしまうためです。これを防ぐには、先頭から削除するのではなく末尾から削除するようにします。

SECTION

089 前後のノードを参照する

ここでは、前後のノードを参照する方法について解説します。

ボタンをクリックすると…

前後のノードを調べてタグであればタグ名、テキストノードであればテキストを表示する

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
```

```
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    前後のノードを参照します。
    <form action="./refnode.cgi" method="get" name="mainForm">
      <input type="button" id="checkButton" value="ノードを参照"><br>
    </form>
    <div id="dNode"><span>ノード1</span><span>ノード2</span>
    <span>ノード3</span></div><div id="result"></div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("checkButton").onclick = function(){
    var base = document.getElementById("dNode"); ―1
    var res = document.getElementById("result");
    var text = checkNode.print(base.firstChild)+"<br>"; ―2
    text += checkNode.print(base.nextSibling)+"<br>"; ―3
    text +=―4
checkNode.print(base.lastChild.previousSibling.firstChild)+"<br>";
    res.innerHTML = text;
  }
}
checkNode = {
  print : function(aNode){
    if (aNode.nodeType == 1){ return aNode.tagName; }
    if (aNode.nodeType == 3) { return aNode.nodeValue; }
    return "---";
  }
}
```

1 …読み出すノードの基準となる領域の情報を読み出す(ID名「dNode」)

2 …最初の子ノードを参照する(<span>)

3 …次のノードを参照する(<div>)

4 …最後の子ノードの1つ前のノードの子ノードを参照する

OnePoint 「previousSibling」「nextSibling」で前後のノードを参照する

1つ前のノードを参照するには「previousSibling」プロパティ、1つ先の「nextSibling」プロパティを使って指定します。これは、現在のノードの前後のノードを示すプロパティで、現在のノードの子ノードを示すわけではない点に注意してください。

090 テキストノードを作成する

ここでは、ボタンをクリックしたときに、現在時刻をテキストノードとして作成する方法について解説します。

ボタンをクリックすると…

現在の時刻がテキストノードとして追加表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; ↩
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css" ↩
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
```

```
  <body>
    テキストノードを作成して現在時刻を表示します。
    <form action="./mknode.cgi" method="get" name="mainForm">
      <input type="button" id="createButton" value="ノードを生成"><br>
    </form>
    <div id="result"></div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("createButton").onclick = function(){
    var dateObj = new Date();
    var h = dateObj.getHours();
    var m = dateObj.getMinutes();
    var s = dateObj.getSeconds();
    var text = document.createTextNode(h+"時"+m+"分"+s+"秒");
    document.getElementById("result").appendChild(text);
  }
}
```

1 …「Date」オブジェクトを生成して現在の時刻を読み出す
2 …テキストノードを生成する
3 …生成したテキストノードを領域に連結してページ上に表示する

OnePoint テキストノードの生成は「document.createTextNode()」を使う

テキストノードを生成するには、「document.createTextNode()」を使います。パラメータに生成するテキストを指定します。テキストノードを生成しただけではページ上には表示されないので、ページ上のノードに連結する必要があります。

サンプルでは現在の時刻をテキストノードとして生成し、<div>タグで指定された領域の末尾に追加しています。

Column コメントノードを生成する

テキストノードやエレメントだけでなくコメントノードも生成することができます。コメントノードを生成するには「createComment()」を使います。パラメータには生成したいコメント内容を指定します。

```
var commentNode = document.createComment("出版社からのコメント");
document.getElementById("result").appendChild(commentNode);
```

SECTION 091

エレメントを作成する

ここでは、エレメントとして<span>タグを作成する方法について解説します。

ボタンをクリックすると…

<span>タグが作成され、そこへ現在の時刻が追加される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
```

```
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    ボタンをクリックすると、spanタグを生成して現在時刻を表示します。
    <form action="./mknode.cgi" method="get" name="mainForm">
      <input type="button" id="createButton" ↩
      value="spanエレメントを生成"><br>
    </form>
    <div id="result"></div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("createButton").onclick = function(){
    var dateObj = new Date();          ┐
    var h = dateObj.getHours();        │ 1
    var m = dateObj.getMinutes();      │
    var s = dateObj.getSeconds();      ┘
    var brTag = document.createElement("br"); ← 2
    var spanTag = document.createElement("span"); ← 3
    spanTag.className = "note"; ← 4
    spanTag.innerHTML = h+"時"+m+"分"+s+"秒"; ← 5
    document.getElementById("result").appendChild(spanTag); ┐ 6
    document.getElementById("result").appendChild(brTag);   ┘
  }
}
```

1 …「Date」オブジェクトを生成して現在の時刻を読み出す

2 …エレメント(<br>)を生成する

3 …エレメント(<span>)を生成する

4 …生成した<span>のスタイルシートクラス名を設定する

5 …表示する内容を<span>に設定する

6 …<span>と
をページ上に出力する

OnePoint エレメントの生成は「document.createElement()」を使う

エレメントを生成するには、「document.createElement()」を使います。パラメータに生成するタグ名を指定します。エレメントを生成しただけではページ上には表示されないので、ページ上のノードに連結する必要があります。

サンプルでは現在の時刻を表示するため、<span>タグを生成し内容として設定後、<div>タグで指定された領域の末尾に追加しています。

092 まとめてノードを追加する

ここでは、大量の乱数を生成し、それらをまとめてノードとして追加する方法について解説します。

ボタンをクリックすると…

乱数がページ上に表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
```

```
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    ボタンをクリックすると、乱数が一括で表示されます。
    <form action="./mknode.cgi" method="get" name="mainForm">
      <input type="button" id="addButton" value="ノードを一括して追加"><br>
    </form>
    <div id="result"></div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("addButton").onclick = function(){
    var dfNode = document.createDocumentFragment(); ―1
    for (var i=0; i<1000; i++){
      var text = document.createTextNode(Math.random() * 6 + 1); ―2
      var divTag = document.createElement("div"); ―3
      divTag.appendChild(text); ―4
      dfNode.appendChild(divTag); ―5
    }
    document.getElementById("result").appendChild(dfNode); ―6
  }
}
```

1 …ドキュメントフラグメントを用意する
2 …乱数を生成しテキストノードとする
3 …エレメント(<div>)を生成する
4 …生成した<div>にテキストノードを追加する
5 …ドキュメントフラグメントに生成したエレメント(テキストノードを追加した<div>タグ)を追加する
6 …これまでに生成したノードをまとめてページ上の領域に追加する

OnePoint 大量のノードを追加する場合は「document.createDocumentFragment()」を使う

DOM(Document Object Model:ドキュメントオブジェクトモデル)でのノードの処理が大量に行われた場合には、処理速度が低下して動作に支障をきたす場合があります。そのような場合にはドキュメントフラグメントを生成し、そこにノードの追加などの処理を行い、最後に一括してページ上に表示することができます。

Break Time

HTMLページ内のコメント内容を取得する

DOMが扱えるブラウザでは、JavaScriptによってHTMLタグやテキストをノードして読み出すことができます。次のサンプルは、コメントノードを読み出してコメント内の文字を表示するプログラムです(エンコードした結果が表示される)。ただし、Safari2では、HTMLタグのコメントノードを読み出すことができないので注意してください。

◉HTMLファイル

```
<div id="photoAlbum">
  <img src="images/DSC0001.jpg" width="240" height="135"><!--
  Sample1 -->
  <img src="images/DSC0042.jpg" width="240" height="135">
  <img src="images/DSC0103.jpg" width="240" height="135">
</div>
```

◉JavaScriptファイル

```
window.onload = function(){
  var imgTag = document.getElementById("photoAlbum").
  getElementsByTagName("img");
  alert(escape(imgTag[0].nextSibling.nodeValue));
}
```

JavaScript Sample - Windows Internet Explorer
G:¥章末コラム¥第6
Live Search
JavaScript Sample
コメントテスト
Windows Internet Explorer
%20Sample1%20
OK
コンピュータ | 保護モード: 無効
100%

Internet Explorer7での実行結果

CHAPTER - 07

スタイルシート

SECTION 093 スタイルシートを制御する方法について

スタイルシートの制御方法

JavaScriptでは、定義されたスタイルシートプロパティ値を参照したり、設定したりする方法がいくつか用意されており、目的に応じて使い分けます。

読み出される値はブラウザによって異なり、16進数の値で返すブラウザもあれば、定義したカラー名(redなど)をそのまま返すブラウザもあるので注意が必要です。そのため、どの方法でスタイルシートプロパティ値を読み出したのか、把握しておく必要があります。

JavaScriptでスタイルシートのプロパティを指定する場合には、「color」や「position」などのようにそのまま指定できるスタイルシートプロパティとそうでないスタイルシートプロパティがあります。たとえば、スタイルシートで背景色を示すプロパティは「background-color」ですが、JavaScriptではこのまま指定することはできません。この場合、プロパティ名から「-」(ハイフン)を削除して直後の文字を大文字にして指定します。「background-color」であれば、「backgroundColor」となります。

▶HTMLタグの「style」属性を利用する方法

HTMLタグの「style」属性に直接指定されたスタイルシートプロパティは「style.プロパティ名」として参照/設定することができます。このアクセス方法は、古いブラウザでも利用することができます。

▶DOMを利用する方法

HTMLタグで「style」属性が指定されていない場合など、「document.defaultView.getComputed Style()」を使って、実際にブラウザに表示されているプロパティ値を読み出す方法もあります。読み出すプロパティ名は、「getPropertyValue()」で指定します。ただし、この方法はDOMが利用できないブラウザや古いブラウザでは利用できないので注意してください。なお、Internet Explorerは、この方法ではなく、HTMLタグオブジェクトの「current Style.getAttribute()」を使ってプロパティ値を読み出します。

サンプルの、このボタンをクリックすると・・・

JavaScript Sample - Windows Internet Explorer
C:¥JavaScript-Sam
Live Search
JavaScript Sample
ページ(P)
ボタンをクリックすると、スタイルシートの情報を表示します。
box1の実際の文字色を表示
box1の外部CSSでの値を表示
box1のcolorのstyle属性値を表示
red
ここのスタイルシートを読み出すことができます
ID名「box1」
コンピュータ | 保護モード: 無効
100%

DOMを利用して、ID名が「box1」の文字色を実際のブラウザ上から取得して表示する

▶外部CSSに直接アクセスする方法

別ファイルで定義されているスタイルシートプロパティ値を読み出すこともできます。その場合、「document.styleSheets.item()」を使います。「item()」のパラメータには、読み出すスタイルシートの順番を指定します。この順番は最初が0番になります。

このスタイルシートから読み出すセレクタを「cssRules.item()」を使って、読み出す位置を指定します。

Internet Explorerの場合は、「rules.item()」を使います。

　外部CSSにアクセスして読み出されるプロパティ値は、画面に表示されている値ではなく、実際にファイル内で定義されている値になるということに注意してください。

二番目のボタンをクリックすると、外部CSSに直接アクセスして、ID名が「box1」のスタイルシートの文字色を取得して表示する

JavaScript Sample - Windows Internet Explorer
ボタンをクリックすると、スタイルシートの情報を表示します。
box1の実際の文字色を表示
box1の外部CSSでの値を表示
box1のcolorのstyle属性値を表示
#2233ff
ここのスタイルシートを読み出すことができます
ID名「box1」

JavaScript Sample - Windows Internet Explorer
ボタンをクリックすると、スタイルシートの情報を表示します。
box1の実際の文字色を表示
box1の外部CSSでの値を表示
box1のcolorのstyle属性値を表示
red
ここのスタイルシートを読み出すことができます
ID名「box1」

一番下のボタンをクリックすると、ID名が「box1」の「style」属性に指定されているスタイルシートから文字色を取得して表示する

SOURCE CODE　HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; ⤶
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css" ⤶
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    ボタンをクリックすると、スタイルシートの情報を表示します。
    <form action="./getcss.cgi" method="get" name="mainForm">
      <input type="button" id="getButton1" ⤶
      value="box1の実際の文字色を表示">
      <input type="button" id="getButton2" ⤶
      value="box1の外部CSSでの値を表示">
      <input type="button" id="getButton3" ⤶
      value="box1のcolorのstyle属性値を表示">
```

```
    </form>
    <div id="result"></div>
    <div id="box1" style="color:red">ここのスタイルシートを読み出すことができます
    </div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("getButton1").onclick = function(){
    var col;
    var divTag = document.getElementById("box1");
    if (document.defaultView){
      col = document.defaultView.getComputedStyle(divTag,
         null).getPropertyValue("color");              ← 1
    }
    if (divTag.currentStyle){
      col = divTag.currentStyle.getAttribute("color");  ← 2
    }
    document.getElementById("result").innerHTML = col;
  }
  document.getElementById("getButton2").onclick = function(){
    var cssValue;
    var divTag = document.getElementById("box2");
    var cssRoot = document.styleSheets.item(0);  ← 3

    if (cssRoot.cssRules){
      cssValue = cssRoot.cssRules.item(3).style.color;  ← 4
    }
    if (cssRoot.rules){
      cssValue = cssRoot.rules.item(3).style.color;  ← 5
    }
    document.getElementById("result").innerHTML = cssValue;
  }
  document.getElementById("getButton3").onclick = function(){
    var divTag = document.getElementById("box1");
    document.getElementById("result").innerHTML =
    divTag.style.color;  ← 6
  }
}
```

1 …画面上に表示されている実際の文字の色を求める(DOMに対応したブラウザの場合)

2 …画面上に表示されている実際の文字の色を求める(Internet Explorerの場合)

3 …定義されているスタイルシートの最初のセレクタを指定する

4 …定義されているスタイルシートの4番目のセレクタの文字色を求める(DOMに対応したブラウザの場合)

5 …定義されているスタイルシートの4番目のセレクタの文字色を求める(Internet Explorerの場合)

6 …タグに指定されているstyle属性での文字色を求める

SOURCE CODE　CSS main.css

```
body {
  background-color:#fff;
  color:#333;
  font-size:80%;
  line-height:1.8;
}
h1 {
  font-size:14pt;
  color:#777;
  padding:0;
  margin:0;
}
form {
  width:300px;
  padding:8px;
  border:1px #ddf solid;
  background-color:#f0f0ff;
}
#box1 { ―1
  width:300px;
  padding:8px;

  color:#2233ff; ―2
  border:1px #fdd solid;
  background-color:#fff0f0;
}
#result {
  width:300px;
  padding:8px;
  border:1px #dfd solid;
  background-color:#f0fff0;
}
```

1 …JavaScriptで取得するセレクタの位置

2 …JavaScriptで取得する文字色

094 文字色と背景色を設定する

ここでは、文字色と背景色を設定する方法について解説します。

SOURCE CODE　HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
```

```
    <meta http-equiv="content-type" content="text/html; 
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css" 
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>カラーシミュレーター</h1>
    <form action="./setcss.cgi" method="get" name="mainForm">
      文字色:<input type="text" name="textColor" 
      id="textColor" value="#fff"><br>
      背景色:<input type="text" name="bgColor" 
      id="bgColor" value="#008"><br>
      <input type="button" id="setButton" value="スタイルを変更">
    </form>
    <div id="result">ここのエリアで背景色と文字色を確認できます。</div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("setButton").onclick = function(){
    var tColor = document.getElementById("textColor").value; —1
    var bColor = document.getElementById("bgColor").value; —2
    document.getElementById("result").style.color = tColor; —3
 4— document.getElementById("result").style.backgroundColor = bColor;
  }
}
```

1 …テキストフィールドに入力されているカラーコード(文字色)を読み出す

2 …テキストフィールドに入力されているカラーコード(背景色)を読み出す

3 …ID名「result」のタグの文字色を設定する

4 …ID名「result」のタグの背景色を設定する

OnePoint 文字色は「color」、背景色は「backgroundColor」で指定する

JavaScriptから画面上のスタイルシートを操作するには、操作したいタグの「style」オブジェクトのプロパティに値を設定します。文字の色は「color」プロパティ、背景色は「backgroundColor」プロパティで指定します。指定できる値はHTMLのカラー名だけでなくスタイルシートで指定できる形式(「rgb(~)」や「#35f」など)も可能です。

SECTION

095 不透明度を設定する

ここでは、不透明度を設定する方法について解説します。

不透明度を入力してボタンをクリックすると…

ボックスの不透明が変更される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
```

```
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>不透明度の確認</h1>
    <form action="./setcss.cgi" method="get" name="mainForm">
      不透明度:<input type="text" name="opac" id="opac"
      value="0.25"><br>
      <input type="button" id="setButton" value="不透明度を変更">
    </form>
    <div id="result">ここのエリアで不透明度を確認できます。</div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("setButton").onclick = function(){
    var n = parseFloat(document.getElementById("opac").value); ―1
    if (window.createPopup){ ―2
      document.getElementById("result").filters.alpha.Opacity = ┐3
      n * 100;                                                  ┘
    }else{
      document.getElementById("result").style.opacity = n; ―4
    }
  }
}
```

1 …テキストフィールドに入力されている不透明度を読み出す

2 …Internet Explorerかどうか判別する

3 …不透明度を設定する(Internet Explorerの場合)

4 …不透明度を設定する(Internet Explorer以外のブラウザ)

OnePoint 不透明度は「opacity」、「filters.alpha.Opacity」で指定する

不透明度が指定できるブラウザは限定されており、Internet Explorer4以降、Safari、Opera9以降、Firefoxが対応しています。また、不透明度の設定はInternet Explorerと、他のブラウザで異なっています。Internet Explorerの場合は、フィルタの「alpha.Opacity」プロパティに0～100までの値を設定します。Internet Explorer以外のブラウザでは、スタイルシートの「opacity」プロパティに0～1.0までの小数値で指定します。

096 現在のスタイル情報を読み出す

ここでは、タグに設定されている実際のスタイルプロパティ値を読み出す方法について解説します。

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>座標と大きさの確認</h1>
    <form action="./readcss.cgi" method="get"
    id="mainForm" name="mainForm">
      <input type="button" id="readButton" value="確認"><br>
    </form>
    <div id="box">
      ここのエリアの座標値が表示されます。<br>
```

```
        <span id="result">結果:</span>
      </div>
  </body>
</html>
```

SOURCE CODE　CSS main.css

```
window.onload = function(){
  document.getElementById("readButton").onclick = function(){
    var x = cssInfo.getValue("box", "left");  ―1
    var y = cssInfo.getValue("box", "top");  ―2
    var w = cssInfo.getValue("box", "width");  ―3
    var h = cssInfo.getValue("box", "height");  ―4
    document.getElementById("result").innerHTML = 
    "座標("+x+","+y+")で大きさは["+w+","+h+"]です";
  }
}
var cssInfo = {
  getValue : function(tagID, propName){
    var divBox = document.getElementById(tagID);
    if (document.defaultView){
      var boxCSS = 
      document.defaultView.getComputedStyle(divBox, null);
      return boxCSS.getPropertyValue(propName);
    }                                                        ―5
    if (divBox.currentStyle){
      return divBox.currentStyle.getAttribute(propName);     ―6
    }
  }
}
```

1 …ID名が「box」のスタイルシートプロパティである「left」の値を求める

2 …ID名が「box」のスタイルシートプロパティである「top」の値を求める

3 …ID名が「box」のスタイルシートプロパティである「width」の値を求める

4 …ID名が「box」のスタイルシートプロパティである「height」の値を求める

5 …指定されたプロパティ名の値を取得し返す(Internet Explorer以外のブラウザ)

6 …指定されたプロパティ名の値を取得し返す(Internet Explorer以外の場合)

OnePoint スタイルシートの値は「getPropertyValue()」、「getAttribute()」で取得する

現在のスタイルシートの値を読み出すには、DOMに対応したブラウザであれば「getPropertyValue()」、Internet Explorerの場合は「getAttribute()」を使って取得します。取得できるのは、実際に表示されている値になります。ただし、座標値などは必ずしも数値を返すわけではなく、「auto」などを返すこともあります。また、数値の場合には単位が付いてくるため、数値として処理したい場合には、「parseInt()」または「parseFloat()」を使って変換する必要があります。

097 スタイルシートクラスを入れ替える

ここでは、タグにスタイルシートクラスが1つだけ設定されている場合にクラスを入れ替える処理について解説します。

ボタンをクリックするとスタイルシートが切り替わります。

クラスをnormalに変更
クラスをcautionに変更

本日は晴天なり

ボタンをクリックすると…

ボックスのスタイルシートクラスが切り替わる

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
```

```
  <body>
    ボタンをクリックするとスタイルシートが切り替わります。
    <form action="./setcss.cgi" method="get" name="mainForm">
      <input type="button" id="setButton1" value="クラスをnormalに変更">
      <input type="button" id="setButton2" value="クラスをcautionに変更">
    </form>
    <div id="result" class="normal">本日は晴天なり</div>
  </body>
</html>
```

SOURCE CODE　JavaScript main.js

```
window.onload = function(){
  document.getElementById("setButton1").onclick = function(){
    document.getElementById("result").className = "normal"; ―1
  }
  document.getElementById("setButton2").onclick = function(){
    document.getElementById("result").className = "caution"; ―2
  }
}
```

1 …ID名が「result」のスタイルシートクラスを「normal」に設定する

2 …ID名が「result」のスタイルシートクラスを「caution」に設定する

OnePoint スタイルシートクラスは「className」プロパティに設定する

タグに任意のスタイルシートクラスを設定するには、「className」プロパティに適用したいクラス名を指定します。スタイルシートクラス名の指定は、半角空白で区切って列記することで複数のスタイルを一括して設定することもできます。スタイルシートクラスが複数設定されており、任意のクラスだけを入れ替える場合には、230ページを参照してください。なお、古いブラウザではスタイルシートクラス名を半角空白で区切って複数指定することはできないので注意してください。

指定されているスタイルシートのクラス名を配列として抽出して調べたい場合には、次のように「split()」を使います。複数クラスが指定されている場合は、半角空白で区切られているので「split()」のパラメータには「" "」として半角空白を指定します。

```
var css = document.getElementById("result").className.split(" ");
```

単純に特定のクラスが適用されているかどうかは、「indexOf()」または「lastIndexOf()」を使います。

```
var css = 
document.getElementById("result").className.indexOf("import");
```

先頭が「mz」で始まるクラス名だけを抽出したい場合には、正規表現（match）を使います。

```
var css = 
document.getElementById("result").className.match(/mz/g);
```

SECTION 098 複数のクラスが指定されている場合にスタイルシートクラスを入れ替える

ここでは、タグにスタイルシートクラスが複数設定されている場合に、特定のクラスだけを入れ替える処理について解説します。

JavaScript Sample - Windows Internet Explorer

ボタンをクリックすると特定のスタイルシートクラスのみ切り替わります。

クラスをnormalに変更
クラスをcautionに変更

ここのエリアの背景色と文字色が変わります。
しかし文字サイズは変わりません。

ボタンをクリックすると…

ボックスの特定のスタイルシートクラスだけが切り替わる

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
```

```
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    ボタンをクリックすると特定のスタイルシートクラスのみ切り替わります。
    <form action="./setcss.cgi" method="get" name="mainForm">
      <input type="button" id="setButton1" value="クラスをnormalに変更">
      <input type="button" id="setButton2" value="クラスをcautionに変更">
    </form>
    <div id="result" class="common normal">
      ここのエリアの背景色と文字色が変わります。<br>
      しかし文字サイズは変わりません。
    </div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("setButton1").onclick = function(){
    var divBox = document.getElementById("result");
    divBox.className =
    divBox.className.replace(/caution/g, "normal");   ← 1
  }
  document.getElementById("setButton2").onclick = function(){
    var divBox = document.getElementById("result");
    divBox.className =
    divBox.className.replace(/normal/g, "caution");   ← 2
  }
}
```

1 …ID名が「result」のスタイルシートクラスに含まれる「caution」クラスだけを「normal」に設定する

2 …ID名が「result」のスタイルシートクラスに含まれる「normal」クラスだけを「caution」に設定する

OnePoint 複数指定されている場合は正規表現を使ってスタイルシートクラスを置換する

スタイルシートクラスが複数指定されている場合に、特定のクラスだけを入れ替えるには、正規表現を利用して置換を行う「replace()」を使います。現在、設定されているスタイルシートクラスを読み出し、特定のクラスだけを置換した後に、再度「className」プロパティに代入することで特定のクラス名だけを入れ替えることができます。

SECTION 099 画像やエレメントの表示/非表示を切り替える

ここでは、画像やエレメントなどブラウザ上に表示されているオブジェクトの表示/非表示を指定する方法について解説します。

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
```

```
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all" id="cssMain">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>更新情報</h1>
    <form action="./setcss.cgi" method="get" name="mainForm">
      <input type="button" id="setButton1" value="表示する">
      <input type="button" id="setButton2" value="表示しない">
    </form>
    <div id="info">
      <img src="images/mark.gif" width="32" height="33" alt="注意">
      本日の更新は特にありません。
    </div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("setButton1").onclick = function(){
    document.getElementById("info").style.visibility = "visible"; ―1
  }
  document.getElementById("setButton2").onclick = function(){
    document.getElementById("info").style.visibility = "hidden"; ―2
  }
}
```

1 …ID名が「info」のスタイルを表示(visible)に設定する

2 …ID名が「info」のスタイルを非表示(hidden)に設定する

OnePoint スタイルシートの「visibility」プロパティで表示/非表示の設定を行う

ブラウザ上に表示されている画像やテキスト、エレメントなどの表示、非表示を行うにはスタイルシートの「visibility」プロパティを操作します。この「visibility」プロパティに文字列として「visible」を指定すると表示され、「hidden」を指定すると表示されなくなります。「visibility」プロパティで「hidden」を設定すると単純に、そのエレメントの領域が表示されなくなります。単純に領域が非表示になるのではなく、以後に続く内容を詰めたい場合には234ページで説明する「display」プロパティを操作します。

SECTION 100

クリックで内容を展開してページ内容を押し下げる

ここでは、エレメントの領域を展開して内容を押し下げる、または内容を詰める方法について解説します。

リンクをクリックすると…

写真が表示され、以後の内容が下に押し下げられる

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all" id="cssMain">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>フォトギャラリー</h1>
    <div id="photoAlbum">
      <div class="photo">
        <a href="images/DSC0001.jpg">写真1[木曽の大橋]</a><br>
        <img src="images/DSC0001.jpg" width="240" height="135">
      </div>
```

```
        <div class="photo">
          <a href="images/DSC0002.jpg">写真2[諏訪湖花火]</a><br>
          <img src="images/DSC0002.jpg" width="240" height="135">
        </div>
        <div class="photo">
          <a href="images/DSC0003.jpg">写真3[高原]</a><br>
          <img src="images/DSC0003.jpg" width="240" height="135">
        </div>
      </div>
    </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  var linkTag =
  document.getElementById("photoAlbum").getElementsByTagName("a");   ← 1
  for (var i=0; i<linkTag.length; i++){ ← 2
    linkTag[i].onclick = function(){ ← 3
      var ptr = this.nextSibling; ← 4
      while((ptr.nodeType != 1) || (ptr.tagName.toLowerCase() !=
      "img")){                                                         ← 5
        ptr = ptr.nextSibling;
      }
      if (ptr.style.display == "") ptr.style.display = "block";
      else ptr.style.display = "";                                     ← 6
      return false;
    }
  }
}
```

1 …ID名が「photoAlbum」のオブジェクト内にある<a>タグの一覧を取得する

2 …取得した<a>タグの数だけ繰り返す

3 …<a>タグに「onclick」イベントを設定する

4 …<a>タグの次のノードを変数「ptr」に入れる

5 …変数「ptr」で示されるノードが<img>タグでない場合には空白を示しているため、次のノードを変数「ptr」に入れる

6 …ノードは<img>タグを示すためスタイルシートの「display」プロパティを調べ表示されていたら「""」、そうでない場合は「"block"」を設定する

OnePoint スタイルシートの「display」プロパティで内容の展開/折りたたみを行う

スタイルシートの「visibility」プロパティでは単純にエレメントの表示/非表示が行われるだけですが、「display」プロパティを使えば、以後に続く内容を押し下げたり詰めたりすることができます。

「display」プロパティに「block」を指定すると内容が(ブロックレベル要素として)表示されます。逆に内容を詰めたい場合には「none」ではなく、「""」を指定します。

SECTION

101 ボタンクリックでCSSファイルを切り替える

ここでは、<link>タグで読み込まれたスタイルシートファイルを切り替える方法について解説します。

ボタンをクリックすると…

指定したスタイルシートファイルが読み込まれ、スタイル表示が切り替わる

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
```

```
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all" id="cssMain">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    ボタンをクリックするとスタイルシートが切り替わります。
    <form action="./setcss.cgi" method="get" name="mainForm">
      <input type="button" id="setButton1" value="main.cssに切替え"><br>
      <input type="button" id="setButton2"
      value="main2.cssに切替え"><br>
    </form>
    <div id="box1">ここにもスタイルシートが設定されています。</div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("setButton1").onclick = function(){
    document.getElementById("cssMain").href = "main.css"; ―1
  }
  document.getElementById("setButton2").onclick = function(){
    document.getElementById("cssMain").href = "main2.css"; ―2
  }
}
```

1 …ID名が「cssMain」の<link>タグのURLを「main.css」に設定する

2 …ID名が「cssMain」の<link>のURLを「main2.css」に設定する

OnePoint <link>タグの「href」属性にスタイルシートファイルのURLを指定する

スタイルシートファイルを切り替えるには、<link>タグの「href」属性に新しく設定するスタイルシートファイルのURLを設定します。<link>タグが複数ある場合には、個別のID名を割り当てておくか、「document.getElementsByTagName("link")」として配列で参照し設定を行うことができます。

Column DOCTYPEによって結果が異なる

Internet Explorerでは、DOCTYPEによって「height」プロパティなど読み出される結果が異なることがあります。この問題は、HTML 4 Transitionalが指定されている場合に発生します。HTML 4 StrictやXHTMLでは発生しません。このDOCTYPEによる不具合は、prototype.jsとscript.aculo.usの組み合わせや、Adobe Spryなどでも発生するため制作時には注意が必要です。

SECTION 102 時間帯別にCSSファイルを切り替える

ここでは、時間帯別にCSSファイルを切り替える方法について解説します。

ページが読み込まれると、時間帯別に異なるスタイルシートファイルが適用される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <link rel="stylesheet" type="text/css"
    href="time0.css" media="all" id="timeCSS">    ― 1
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>ようこそ</h1>
    <div id="box">ゆっくりしていってください。</div>
  </body>
</html>
```

1 …操作する<link>タグにID名を指定する

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  var cssFile = document.getElementById("timeCSS"); ― 1
  var dateObj = new Date(); ― 2
  var h = dateObj.getHours(); ― 3
  cssFile.href = "css/time"+h+".css"; ― 4
}
```

1 …ID名が「timeCSS」のエレメント(<link>タグ)を指定する

2 …日付オブジェクトを生成する

3 …時間を読み出す(読み出した時間は24時間制になるため0〜23の値になる)

4 …スタイルシートファイルの名前と時間を連結したスタイルシートファイルを読み込ませる

OnePoint スタイルシートファイル名を工夫して時間を示す値を追加しておく

読み込むスタイルシートファイルを時間帯別に切り替えるには、<link>タグにID名を指定しておきます。次に読み込むスタイルシートファイル名に工夫をしておきます。サンプルでは、「time」と時間を示す値の組み合わせにしてあります。たとえば、6時であれば「time6.css」、22時であれば「time22.css」というファイル名になります(24時間分用意しておく)。このファイル名(URL)を<link>タグの「href」属性に設定することで時間帯別に異なるスタイルシートを適用することができます。

SECTION

103 スタイルシートを無効にする

ここでは、スタイルシートを無効にする方法について解説します。

ボタンをクリックすると…

スタイルシートの有効/無効を切り替えることができる

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all" id="cssMain">
```

```
    <link rel="stylesheet" type="text/css" href="sub.css"
    media="all" id="cssSub">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    ボタンをクリックすると、スタイルシートの有効/無効を切り替えることができます。
    <form action="./setcss.cgi" method="get" name="mainForm">
      <input type="button" id="setButton1" value="main.cssを無効"><br>
      <input type="button" id="setButton2" value="main.cssを有効"><br>
      <input type="button" id="setButton3" value="sub.cssを無効"><br>
      <input type="button" id="setButton4" value="sub.cssを有効"><br>
    </form>
    <div id="box1">ここはmain.cssでスタイルが定義されています。</div>
    <div id="box2">ここはsub.cssでスタイルが定義されています。</div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("setButton1").onclick = function(){
    document.getElementById("cssMain").disabled = true; —1
  }
  document.getElementById("setButton2").onclick = function(){
    document.getElementById("cssMain").disabled = false; —2
  }
  document.getElementById("setButton3").onclick = function(){
    document.getElementById("cssSub").disabled = true; —3
  }
  document.getElementById("setButton4").onclick = function(){
    document.getElementById("cssSub").disabled = false; —4
  }
}
```

1 …ID名「cssMain」のエレメントのスタイルシートを無効にする
2 …ID名「cssMain」のエレメントのスタイルシートを有効にする
3 …ID名「cssSub」のエレメントのスタイルシートを無効にする
4 …ID名「cssSub」のエレメントのスタイルシートを有効にする

OnePoint 「disabled」プロパティで有効/無効を指定する

読み込まれたスタイルシートファイルを有効にしたり、無効にしたりする場合は「disabled」プロパティを操作します。「disabled」プロパティに「true」を設定すると、スタイルシートは無効になります。「false」を設定するとスタイルシートは有効になります。

Break Time

ローカル変数のスコープ

JavaScriptのローカル変数は、関数内でのみ有効です。他の言語ではブロック内でのみ変数が有効となることが多いのですが、JavaScriptでは関数内全体に変数の影響が及んでしまいます。Firefox2などでは「let」によりブロック内のみ有効とすることができますが、他のブラウザではサポートされていないため、別の方法を使用することになります。

関数内で変数の影響があるわけですから、次のサンプルのように無名関数/匿名関数でくくってしまえば、ある程度までは問題を解決することができます。

```
function test1(){
  var n = 0, i = 999;
  for (var i=0; i<10; i++) { /* 処理 */ }
  alert(i);
}
function test2(){
  var n=0, i=999;
  (function (){
    for (var i=0; i<10; i++) { /* 処理 */ }
  });
  alert(i);
}
test1();
test2();
```

上記のサンプルを実行すると、次のようにアラートが2回、表示されます。最初のアラートは関数「test1()」の実行結果、2番目のアラートは関数「test2()」の実行結果になります。「test1()」では変数「i」に999代入していますが、「for」命令でも変数「i」を利用しているため、最終的な変数「i」の値は「10」になります。一方、「test2()」では、「for」命令を無名関数でくくっているため、変数「i」の値は「999」と表示されます。

Windows Internet Explorer
10
OK

Windows Internet Explorer
999
OK

CHAPTER - 08

イベント

104 イベントの発生したタイミングでイベントオブジェクトから情報を取得する

ここでは、イベント発生時にイベントオブジェクトから情報を取得して、ページ上に表示する方法について解説します。

JavaScript Sample - Windows Internet Explorer

ボタンをクリックすると、クリックした座標とオブジェクトのIDが表示されます。

クリックしてください

結果：

ボタンをクリックすると…

156,88
checkButton

クリックした座標値とイベントが発生したオブジェクトのID名が表示される

SOURCE CODE　HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; 
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css" 
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    ボタンをクリックすると、クリックした座標とオブジェクトのIDが表示されます。
    <form action="./event.cgi" method="get" name="mainForm">
      <input type="button" id="checkButton" value="クリックしてください"><br>
    </form>
```

```
    <div id="result">結果:</div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("checkButton").onclick = function(evt){ ─1
    if (!evt) { ─2
      document.getElementById("result").innerHTML =
      event.x +","+ event.y+"<br>";                              ─3
      document.getElementById("result").innerHTML +=
      event.srcElement.id;
    }else{
      document.getElementById("result").innerHTML =
      evt.pageX +","+ evt.pageY+"<br>";                          ─4
      document.getElementById("result").innerHTML += evt.target.id;
    }
  }
}
```

1 …イベントが発生するとイベントハンドラにパラメータとしてイベントオブジェクトが渡される(Internet Explorer以外)

2 …イベントハンドラにイベントオブジェクトが渡されたかどうか調べる(渡されていなければInternet Explorer、渡されていればInternet Explorer以外のブラウザとする)

3 …Internet Explorerの場合は「event」オブジェクトに情報が入っているので「x」プロパティと「y」プロパティから座標値を取り出す

4 …Firefox、Safari、Operaなどではイベントハンドラに渡されたイベントオブジェクトに情報が入っているので「pageX」プロパティと「pageY」プロパティから情報を取り出す

OnePoint Internet Explorerとそれ以外で異なるイベントの扱いに注意する

イベントオブジェクトは、Internet Explorerでは「event」オブジェクト、他ブラウザではイベントハンドラに渡されます。「event」オブジェクト自体はSafari2やOperaでも存在し、Internet Explorer同様にアクセスすることができます。ただし、Firefoxでは「event」オブジェクトではなく、イベントハンドラにイベントオブジェクトを渡すようになってます。標準的な仕様に合わせるのであればFirefoxになりますが、ブラウザごとに対応が異なるため、サンプルのようにイベントオブジェクトの有無に応じてInternet Explorerと他ブラウザで処理を分ける必要があります。

また、イベントオブジェクトで扱われる各種プロパティについても互換性があるプロパティは、かなり限られるため、扱いには注意が必要です。特にイベント発生時の座標値に関してはブラウザごと(Firefox、Safari2、Internet Explorer)に異なるため注意が必要です。

SECTION 105

イベントの設定と解除をボタン操作で切り替える

ここでは、ボタンのクリックによってイベントの設定/解除を切り替える方法について解説します。

JavaScript Sample - Windows Internet Explorer

ボタン
上のボタンにイベントを設定
上のボタンからイベントを解除

一番上のボタンにイベントを設定/解除することができる

Windows Internet Explorer

クリックされました

OK

イベントが設定されているときに一番上のボタンをクリックするとアラートが表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <form action="./event.cgi" method="get" name="mainForm">
      <input type="button" id="checkButton" value="ボタン"><br>
      <input type="button" id="setButton" value="上のボタンにイベントを設定">
      <input type="button" id="clrButton" value="上のボタンからイベントを解除">
    </form>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("setButton").onclick = function(){
    myEvent.set();
  }
  document.getElementById("clrButton").onclick = function(){
```

```
    myEvent.clear();
  }
}
var myEvent = {
  set : function(){
    var divObj = document.getElementById("checkButton");
    if (divObj.addEventListener) divObj.addEventListener
    ("click", myEvent.message, false);                    ←1
    if (divObj.attachEvent) divObj.attachEvent
    ("onclick", myEvent.message);                         ←2
  },
  clear : function(){
    var divObj = document.getElementById("checkButton");
    if (divObj.removeEventListener) divObj.removeEventListener
    ("click", myEvent.message, false);                    ←3
    if (divObj.detachEvent) divObj.detachEvent
    ("onclick", myEvent.message);                         ←4
  },
  message : function(){
    alert("クリックされました");
  }
}
```

1 …クリックイベントを追加する(Internet Explorer以外の標準仕様に対応したブラウザ)
2 …クリックイベントを追加する(Internet Explorerの場合)
3 …設定したクリックイベントを削除する(Internet Explorer以外の標準仕様に対応したブラウザ)
4 …設定したクリックイベントを削除する(Internet Explorerの場合)

OnePoint イベントを追加するには「addEventListener()」「attachEvent()」を使う

JavaScriptでは、旧来の方法でイベントを設定する方法と、DOM(Level 2)でのイベントを設定する方法があります。旧来の方法では「オブジェクト名.イベント名=イベントハンドラ」としてイベント発生時の処理を指定します。この方法では開発者が複数いる場合や複数のライブラリを使用した場合、後から指定したイベントで上書きされてしまい、それ以前に設定されていたイベントハンドラは消されてしまって実行されなくなってしまいます。

これに対してDOMでのイベント設定は、イベント処理を複数ストックする(キュー)ようになっています。つまり、複数のクリックイベントを設定しても、以前のイベント設定は消されずに期待通り動作します。このようにするとライブラリなどを利用した場合でも、イベント発生時の不具合を軽減することができます。

イベントを追加するには「addEventListener()」を使います。ただし、Internet Explorerでは利用できないため、「attachEvent()」で代用します。「addEventListener()」と「attachEvent()」の動作は似ていますが、イベント発生時の動作に互換性がないのでイベントバブルアップ(イベントが伝達される処理)などの処理には注意が必要です。

「addEventListener()」「attachEvent()」で設定したイベントを削除するには、「removeEventListener()」「detachEvent()」を使います。パラメータにはイベントを追加したときと同じパラメータを指定します。パラメータが異なると、違うイベント設定とみなされ、削除されません。

SECTION 106 ページが読み込まれたタイミングで処理を行う

ここでは、ページが読み込まれたタイミングで自動的に処理を行う方法について説明します。

ページが読み込まれると、ランダムに画像が1枚表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; 
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css" 
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>「私のひとり写真館」へようこそ！</h1>
    <div id="info">お気に入り写真の中の1枚をランダムに公開中！<br></div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){ ―1
  var imageURL =
  ["images/DSC0001.jpg","images/DSC0002.jpg","images/DSC0003.jpg"];
  var imgTag = document.createElement("img");
  var n = Math.floor(Math.random()*imageURL.length);
  imgTag.src = imageURL[n];
  document.getElementById("info").appendChild(imgTag);
}
```

1 …ページが読み込まれると「onLoad」イベントが発生する(スクリプトでは「window.onload」でイベント発生時の処理を指定する)

2 …イベントが発生した時の処理を記述する

OnePoint 「window.onload」でページ読み込み時の処理を指定する

旧来、ページが読み込まれた場合の処理は、<body>タグに「onLoad」イベントを記述していました。たとえば、「<body onLoad="alert(1)">」とすると、ページが読み込まれた後にアラートダイアログが表示されます。しかし、この方法ではHTML文書とプログラムが分離されていないため、分業が難しくなります。そこで、プログラム側でページ読み込み時のイベントハンドラを指定するようにします。

ページが読み込まれた場合に処理を行うには、「window.onload」に実行する処理を記述します。イベント発生時に読み出す関数を指定するのが通例です。サンプルのように無名関数/匿名関数を指定することもできます。

「window.onload」は、単純に「onload」と書いても同じように動作します。これはJavaScriptでは「window」オブジェクトの表記を省略することができるためです。DOMが扱えるブラウザではこのような表記ではなく、「addEventListener()」を使ってイベントを設定する方がよいでしょう。これはライブラリなどで「window.onload」が定義されていれば、その処理を上書きしてしまうためです。また、Intenet Explorerの場合は「addEventListener()」をサポートしていませんが、代わりに「attachEvent()」を使ってイベント処理を設定することができます。

Column 旧来の方法でのイベント削除方法

旧来の方法でのイベント削除方法は、イベントハンドラに「null」を設定することで行います。

```
document.body.onclick = null;
```

また、null以外でも次のように空の関数を指定する方法もあります。

```
document.body.onclick = function(){};
```

SECTION 107

画像やエレメントなどをドラッグできるようにする

ここでは、script.aculo.usライブラリを使って、画像やエレメントなどをドラッグできるようにする方法について解説します。

さまざまなドラッグを実現しています。
ここは普通にドラッグできます
ここはドラッグすると元の位置に戻ります
ドラッグできます

ID名「dragBox1」
ID名「dragBox2」
ID名「dragImage」

●ID名「dragBox1」をドラッグした場合

ドラッグすることができる

●ID名「dragBox2」をドラッグした場合

ドラッグすると…

元の位置に戻って行く

●ID名「dragImage」をドラッグした場合

グリッドに沿ってドラッグさせることができる

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; ↩
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css" ↩
    media="all">
    <script type="text/javascript" src="prototype.js"></script>
    <script type="text/javascript" src="scriptaculous.js"></script>
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    さまざまなドラッグを実現しています。
    <div id="dragBox1">ここは普通にドラッグできます</div>
    <div id="dragBox2">ここはドラッグすると元の位置に戻ります</div>   ← 1
    <img src="images/icon.gif" id="dragImage">
  </body>
</html>
```

1 …ドラッグ対象のエレメントや画像にはID名を指定する

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  new Draggable("dragBox1"); ← 1
  new Draggable("dragBox2", { revert:true } ); ← 2
  new Draggable("dragImage", { snap:function(x,y){
    return [Math.floor(x / 20)*20, Math.floor(y / 20)*20]; }}); ← 3
}
```

1 …ID名「dragBox1」は普通にドラッグできるように設定する

2 …ID名「dragBox2」はドラッグ後に元の位置に戻るように設定する

3 …ID名「dragImage」は20ピクセルごとに移動するように設定する

OnePoint ドラッグを実現するにはscript.aculo.usライブラリの「new Draggable()」を使う

ページ上のエレメントや画像をドラッグできるようにするには、script.aculo.usライブラリを使うと便利です。script.aculo.usでは、「Draggable()」にドラッグ対象となるエレメントのID名またはオブジェクトを指定するだけでドラッグすることが可能になります。

オプションを指定することで、ドラッグ時の処理を設定することができます。オプションは「Draggable()」の2番目のパラメータで指定します。「revert」オプションで「true」を指定すると、ドラッグ後に元の位置に戻ります。「snap」を指定すると、関数の戻り値に応じた座標にスナップするようにドラッグできるようになります。

押されたキーのキーコードを取得する

ここでは、押されたキーのキーコードを取得して、どのキーが押されたかを表示する方法について解説します。

JavaScript Sample - Windows Internet Explorer
押されたキーのキーコードは？
65 (A)
66 (B)
67 (C)
68 (D)
69 (E)
70 (F)

キーを押すと、押されたキーコードと対応する文字が表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; ⤶
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css" ⤶
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    押されたキーのキーコードは?
    <div id="result"></div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  window.document.onkeydown = function(evt){
    if (evt){
      var kc = evt.keyCode;  ―1
    }else{
      var kc = event.keyCode;  ―2
    }
    var chr = String.fromCharCode(kc);
    document.getElementById("result").innerHTML +=
    kc+" ("+chr+")<br>";
  }
}
```

1 …Internet Explorer以外の場合のキー入力データ処理を行う

2 …Internet Explorerの場合のキー入力データ処理を行う

3 …入力されたキーコードを「String.fromCharCode()」を使って文字に変換する

OnePoint キーコードは「keyCode」プロパティで取得する

キーコードは、イベントオブジェクトの「keyCode」プロパティで取得することができます。このプロパティはInternet ExplorerやFirefoxなどの他のブラウザでも同じですが、返されるキーコードは異なる場合があります。また、キーボードの種類や設定によっても変わることがあります。

発生したイベントが「keyup」「keydown」なのか「keypress」なのかによっても返されるキーコードの値が異なるので注意が必要です。

Column イベントが発生した時刻

FirefoxとSafariでは、イベントが発生した時刻を取得することできます。イベント発生時の時刻は、イベントオブジェクトの「timeStamp」プロパティに格納されます。

```
window.onload = function(){
  window.document.onkeydown = function(evt){
    var tms = "";
    if (evt){
      var kc = evt.keyCode;
      tms = evt.timeStamp;
    }else{
      var kc = event.keyCode;
    }
    var chr = String.fromCharCode(kc);
    document.getElementById("result").innerHTML +=
    kc+" ("+chr+" : " + tms +")<br>";
  }
}
```

SECTION 109

現在のマウスカーソルの座標を取得する

ここでは、ブラウザ上にあるマウスカーソルの座標を取得して、その座標をリアルタイムにページ上に表示する方法について解説します。

JavaScript Sample - Windows Internet Explorer

現在のマウスカーソルの座標は？

325,170

マウスカーソルを移動させると、ページ上にマウスカーソルの座標が表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    現在のマウスカーソルの座標は?
    <div id="result">---</div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  window.document.onmousemove = function(evt){
    if (evt){
      var x = evt.pageX; —1
      var y = evt.pageY; —2
    }else{
      var x = event.x + document.body.scrollLeft; —3
      var y = event.y + document.body.scrollTop; —4
    }
    document.getElementById("result").innerHTML = x+","+y;
  }
}
```

1 …Internet Explorer以外でのマウスカーソルのX座標を取得する

2 …Internet Explorer以外でのマウスカーソルのY座標を取得する

3 …Internet ExplorerでのマウスカーソルのX座標+スクロールX座標を取得する

4 …Internet ExplorerでのマウスカーソルのY座標+スクロールY座標を取得する

OnePoint 座標を取得するプロパティはブラウザごとに異なるので注意が必要

マウスカーソルの座標値は、イベントオブジェクトの「x」プロパティと「y」プロパティ、または「pageX」プロパティと「pageY」プロパティで取得することができます。Internet Explorerでは「x」「y」プロパティで座標値を取得できますが、ページがスクロールした場合にはページ上の座標値ではなくブラウザ上の座標値になってしまうため、ページ上の座標値を求める場合にはスクロール量を加算します。横方向のスクロール量は「document.body.scrollLeft」、縦方向のスクロール量は「document.body.scrollTop」となります。

Internet Explorer以外のブラウザではイベントオブジェクトの「pageX」「pageY」プロパティで座標値を読み出すことができます。なお、Safari2の場合は、読み出される「clientX」「clientY」の座標値が他のブラウザと異なるので注意が必要です。

Column Internet Explorerでのエレメント関連プロパティ

エレメントのマージンや枠幅などを示すプロパティは、数多くあります。Internet Explorerの場合、次のURLに関連図が用意されています。

URL http://msdn.microsoft.com/library/ja/default.asp?url=/library/ja/jpisdk/dhtml/measure/measuring.asp

SECTION 110 マウスカーソルを重ねたときに表示されている文字列を入れ替える

ここでは、マウスカーソルを重ねたタイミングで、そこに表示されているテキストを入れ替える方法について解説します。

JavaScript Sample - Windows Internet Explorer

オススメ情報

※マウスカーソルを重ねると、オススメ商品が表示されます。

パソコン情報
デジカメ情報
プリンタ情報

JavaScript Sample - Windows Internet Explorer

オススメ情報

※マウスカーソルを重ねると、オススメ商品が表示されます。

パソコン情報
Nikonがオススメです
プリンタ情報

マウスカーソルを文字の上に重ねると、文字が入れ替わる

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; 
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css" 
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>オススメ情報</h1>
    ※マウスカーソルを重ねると、オススメ商品が表示されます。
```

```
    <div id="info1">パソコン情報</div>
    <div id="info2">デジカメ情報</div>
    <div id="info3">プリンタ情報</div>
  </body>
</html>
```

1 …テキストを入れ替える領域にはID名を指定する

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  var swapText = {
    "info1":["Macがオススメです"],
    "info2":["Nikonがオススメです"],
    "info3":["EPSONがオススメです"]
  }
  for (i in swapText){
    swapText[i][1] = document.getElementById(i).innerHTML;
    document.getElementById(i).onmouseover = function(){
      this.innerHTML = swapText[this.id][0];
    }
    document.getElementById(i).onmouseout = function(){
      this.innerHTML = swapText[this.id][1];
    }
  }
}
```

1 …入れ替えるテキストのID名と入れ替える内容をペアにして用意する
2 …入れ替えるテキストの数だけ繰り返す
3 …最初の状態で表示されている内容を配列変数に入れる
4 …マウスカーソルが重なったときにテキストを入れ替える
5 …マウスカーソルが離れたら最初のテキストに戻す

OnePoint ID名と表示内容をペアにして変数に入れておく

マウスオーバーでテキストを入れ替えるには、入れ替えるテキストをあらかじめ連想配列（ハッシュ）に入れておきます。連想配列には、入れ替える領域のID名と入れ替えるテキストをペアにして設定しておきます。

マウスイベントが発生したら、自分自身のIDを読み出し、そのIDをキーにして連想配列から入れ替えるテキストを読み出します。自分自身のIDは「this.id」として読み出すことができます。

SECTION 111 イベントが発生したオブジェクト(タグ)により動作を変える

ここでは、イベントが発生したオブジェクト(タグ)により動作を変える方法について解説します。

JavaScript Sample - Windows Internet Explorer

いろいろなところをクリックしよう!

ボタン

結果:

ボタンがクリックされました

見出しがクリックされました

フォームがクリックされました

ページ上やボタン上、領域上でクリックすると、どこがクリックされたかが表示される

SOURCE CODE　HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; 
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css" 
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
```

```
  <body>
    <h1>いろいろなところをクリックしよう!</h1>
    <form action="./event.cgi" method="get" name="mainForm">
      <input type="button" id="checkButton" value="ボタン"><br>
    </form>
    <div id="result">結果:</div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  window.document.onclick = function(evt){
    if (!evt) {
      var eType = event.srcElement.tagName; ―1
    }else{
      var eType = evt.target.tagName; ―2
    }
    eType = eType.toLowerCase();
    var text = "";
    switch(eType) {
      case "h1" : text = "見出しがクリックされました";
                  break;
      case "input" : text = "ボタンがクリックされました"; ―3
                  break;
      case "form" : text = "フォームがクリックされました";
                  break;
    }
    document.getElementById("result").innerHTML = text;
  }
}
```

1 …クリックされた場所のタグ名を取得する(Internet Explorerの場合)

2 …クリックされた場所のタグ名を取得する(Internet Explorer以外のブラウザの場合)

3 …イベントが発生した場所のタグ名に応じて表示するメッセージを変える

OnePoint イベント発生時のターゲットを取得する

イベントが発生した場所に応じて処理を変えるには、どのタグ上でイベントが発生したかを求めますが、Internet Explorerと他のブラウザで取得方法が異なっています。Internet Explorerでは、「event」オブジェクトの「srcElement」プロパティにイベントが発生した場所のタグ情報が格納されます。FirefoxやSafari、Operaなどではイベントオブジェクトの「target」プロパティにイベントが発生した場所のタグ情報が格納されます。

タグ名は「tagName」プロパティで取得することができます。サンプルでは「switch...case」で取得したタグ名ごとに処理を分けて異なるテキストをページ上に表示しています。

Break Time

まだ生成されていないオブジェクトにアクセスしてエラー

JavaScriptプログラムは、HTMLファイルが読み込まれた後に実行されるわけではありません。HTML文書内に<script>タグがあるとJavaScriptファイルを読み込んで実行し、その後に続くHTML文書の構築処理を行います。つまり次のようなサンプルプログラムでは、まだHTMLファイルに記述された<div>タグが生成されていないためエラーになります。

◉HTMLファイル

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js?sub1.js,sub2.js">
    </script>
  </head>
  <body>
    <h1>未生成のオブジェクトへのアクセス</h1>
    <div id="result"></div>
  </body>
</html>
```

◉JavaScriptファイル(実行するとエラー)

```
document.getElementById("result").innerHTML = "OK";
```

HTMLファイルが読み込まれたら処理するためには、ページが完全に読み込まれて構築されたことを示す「onload」イベントが発生したら処理を行うようにします。つまり上記のサンプルは、次のように記述しないと正しく動作しません。

```
window.onload = function(){
  document.getElementById("result").innerHTML = "OK";
}
```

CHAPTER - 09

文字列

112 テキストフィールドに入力した文字列から1文字だけ抜き出す

ここでは、文字列の中から先頭の1文字だけを抽出する方法について解説します。

JavaScript Sample - Windows Internet Explorer

文字：りんご
先頭のみ抽出

結果：

ボタンをクリックすると…

JavaScript Sample - Windows Internet Explorer

文字：りんご
先頭のみ抽出

先頭の文字は「り」です

入力された文字列の先頭の1文字が表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
```

```
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <form action="./check.cgi" method="get" name="mainForm">
      文字:<input type="text" name="userName"
      id="userName" value="りんご">
      <input type="button" id="checkButton" value="先頭のみ抽出">
    </form>
    <div id="result">結果:</div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("checkButton").onclick = function(){
    var uName = document.getElementById("userName").value; ―1
    var c = uName.charAt(0); ―2
    document.getElementById("result").innerHTML =
    "先頭の文字は「"+c+"」です";
  }
}
```

1 …テキストフィールドの内容を読み出す
2 …「charAt(0)」として最初の文字を抽出する

OnePoint 文字列から1文字抜き出すには「charAt()」を使う

文字列から任意の位置の1文字を抜き出すには、「charAt()」を使います。パラメータには抜き出したい文字の位置を指定します。文字の位置は、先頭の文字が「0」となる点に注意してください。2番目の文字を抜き出したい場合は、「charAt(1)」となります。

Column 末尾の文字を抜き出すには

先頭の文字ではなく、末尾の1文字を抜き出したい場合は、文字列の長さから「1」を引いた値を指定します。サンプルの場合、次のようにすると、末尾の1文字を抜き出すことができます。

```
var c = uName.charAt(uName.length-1);
```

SECTION

113 指定範囲の文字列を抜き出す

ここでは、文字列の4文字目～7文字目の文字列を抽出する方法について解説します。

住所: 東京都千代田区飯田橋 抽出

結果:

ボタンをクリックすると…

住所: 東京都千代田区飯田橋 抽出

4～7文字は「千代田区」です

入力された文字列の4～7番目の文字が表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
```

```
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <form action="./check.cgi" method="get" name="mainForm">
      住所:<input type="text" name="userAddress"
      id="userAddress" value="東京都千代田区飯田橋">
      <input type="button" id="checkButton" value="抽出">
    </form>
    <div id="result">結果:</div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("checkButton").onclick = function(){
    var uName = document.getElementById("userName").value; ―1
    var str = uName.substring(3,7); ―2
    document.getElementById("result").innerHTML =
    "4～7文字は「"+str+"」です";
  }
}
```

1 …テキストフィールドの内容を読み出す

2 …「substring(3,7)」として最初の4～7文字目を抽出する

OnePoint 指定範囲の文字列を抜き出すには「substring()」を使う

文字列から任意の範囲の文字列を抜き出すには、「substring()」を使います。パラメータは2つあり、抜き出す範囲の開始位置と終了位置を指定します。抜き出す文字の位置は先頭の文字が「0」になるので注意が必要です。「substring(0,1)」とすると先頭の1文字だけ、「substring(0,2)」とすると先頭から2文字が抜き出されます。

抜き出す文字列を末尾から指定したい場合には、「substring()」ではなく、「slice()」を使うと便利です。「slice()」はパラメータに負数を指定することができ、「−1」とすると末尾の文字を指定したことになります。「slice(3,−2)」とすると、先頭から数えて4文字目から末尾から数えて3文字までが抜き出されます。

SECTION 114 指定位置から指定された文字数だけ抜き出す

ここでは、文字列の先頭から6文字分を抽出する方法について解説します。

JavaScript Sample - Windows Internet Explorer

住所: 新潟県新潟市西名目所 抽出

結果:

ボタンをクリックすると…

JavaScript Sample - Windows Internet Explorer

住所: 新潟県新潟市西名目所 抽出

先頭から6文字は「新潟県新潟市」です

入力された文字列の先頭6文字分が表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
```

```
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <form action="./check.cgi" method="get" name="mainForm">
      住所:<input type="text" name="userAddress" id="userAddress"
      value="新潟県新潟市西名目所">
      <input type="button" id="checkButton" value="抽出">
    </form>
    <div id="result">結果:</div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("checkButton").onclick = function(){
    var uAddress = document.getElementById("userAddress").value; —1
    var str = uAddress.substr(0,6); —2
    document.getElementById("result").innerHTML =
    "先頭から6文字は「"+str+"」です";
  }
}
```

1 …テキストフィールドの内容を読み出す

2 …「substr(0,6)」として最初から6文字分を抽出する

OnePoint 指定された数の文字列を抜き出すには「substr()」を使う

指定位置から任意の数の文字を抜き出すには「substr()」を使います。パラメータは2つあり、最初が抜き出す文字の位置、次が抜き出す文字数になります。抜き出す文字の位置は「0」から始まる点に注意してください。「substr(0,2)」とすると、最初の2文字が抜き出されます。抜き出す文字数の方が多い場合には、自動的に文字列の末尾までが抜き出されます。

また、末尾から抜き出したい場合は、「substr()」の最初のパラメータに負数を指定します。「−3」とすると、末尾から3文字目が抜き出す開始位置となります。

SECTION 115 文字列を連結する

ここでは、2つのテキストフィールドに入力された文字列を連結する方法について解説します。

ボタンをクリックすると…

入力された文字列が
連結される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
```

```
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <form action="./check.cgi" method="get" name="mainForm">
      姓:<input type="text" name="uName1" id="uName1" value="山田"><br>
      名:<input type="text" name="uName2" id="uName2" value="次郎">
      <input type="button" id="checkButton" value="連結する">
    </form>
    <div id="result">結果:</div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("checkButton").onclick = function(){
    var str1 = document.getElementById("uName1").value;  ─ 1
    var str2 = document.getElementById("uName2").value;
    var str = str1 + str2; ─ 2
    document.getElementById("result").innerHTML = "氏名は「"+str+"」です";
  }
}
```

1 …テキストフィールドの内容を読み出す

2 …読み出した内容を「+」記号を使って連結する

OnePoint 文字列の連結には+記号を使う

文字列を連結するには、「+」記号を使います。「+」記号は数値演算にも使用されますが、JavaScriptでは数値と文字列、文字列と文字列を「+」記号で加算した場合は文字列として結果を返します。「+」記号での処理は左から行われます。その際、加算/連結するデータの型によって処理が変わることになります。たとえば、「2+3+"KF"」の結果は文字列の「5KF」となります。これは最初に数値として「2+3」が処理され、次に「5+"KF"」となるためです。最初から文字列として連結したい場合には、最初に「""」を付加し、「""+2+3+"KF"」のようにします。

SECTION 116

文字列の中から特定の文字を削除する

ここでは、文字列の中から「a」の文字を削除する方法について解説します。

JavaScript Sample - Windows Internet Explorer
C:¥JavaScript-Sa
Live Search
JavaScript Sample
文字: apartment
aの文字を削除する
結果:
コンピュータ | 保護モード: 無効

ボタンをクリックすると…

JavaScript Sample - Windows Internet Explorer
C:¥JavaScript-Sa
Live Search
JavaScript Sample
文字: apartment
aの文字を削除する
削除した結果は「prtment」です
コンピュータ | 保護モード: 無効
100%

入力された文字列から「a」の文字を削除した結果が表示される

SOURCE CODE　HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
```

```
  <body>
    <form action="./check.cgi" method="get" name="mainForm">
      文字:<input type="text" name="text1" id="text1"
      value="apartment">
      <input type="button" id="checkButton" value="aの文字を削除する">
    </form>
    <div id="result">結果:</div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("checkButton").onclick = function(){
    var uText = document.getElementById("text1").value; —1
    var str = uText.split("a").join(""); —2
    document.getElementById("result").innerHTML =
    "削除した結果は「"+str+"」です";
  }
}
```

1 …テキストフィールドの内容を読み出す

2 …読み出した内容から「split()」を使って「a」の文字を削除した後に「join()」を使って再度連結する

OnePoint 「split()」と「join()」を組み合わせて特定の文字列を削除する

文字列から特定の文字列を削除するには、「spilt()」と「join()」を使います。まず、「split()」を使って削除する文字列を区切りとして文字列を分解します。この時点で削除された文字以外が配列として生成されます。生成された配列を「join()」を使って連結することで、文字列の削除を行うことができます。

Column 文字列を指定するのに「"」と「'」のどちらを使えばよい?

JavaScriptでは文字列を指定する際に「"」(ダブルクォーテーション)または「'」(シングルクォーテーション)で囲みます。他のプログラミング言語の中には「"」と「'」で囲んだ場合に異なる処理が行われるプログラミング言語がありますが、JavaScriptではどちらも同じように処理されます。

文字列内に「"」が入る場合には「'」で文字列を囲みます。逆に文字列内に「'」が含まれる場合には「"」で囲みます。実際のサンプルでは、次のようになります。

```
str1 = "Don't";
str2 = 'Double quote is "';
```

通常は、どちらでも問題ありませんが、Ajaxなどで後付けでプログラムをダイナミックに読み込み処理する場合には、どちらかに統一しておくのが安全です。統一するのであれば、「'」で囲むようにしておくと後々よいでしょう。

SECTION 117

指定した文字列が存在するか調べる

ここでは、テキストフィールド内の文字列に「県」という文字が存在するかどうかを調べる方法について解説します。

JavaScript Sample - Windows Internet Explorer

検索対象文字列: 長野県
検索したい文字: 県
検索する

結果:

ボタンをクリックすると…

検索対象文字列: 長野県
検索したい文字: 県
検索する

3文字目で見つかりました

検索した文字列の最初の位置が表示される

SOURCE CODE　HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
```

```
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <form action="./check.cgi" method="get" name="mainForm">
      検索対象文字列:<input type="text" name="targetText" 
      id="targetText" value="長野県"><br>
      検索したい文字:<input type="text" name="searchText" 
      id="searchText" value="県"><br>
      <input type="button" id="checkButton" value="検索する">
    </form>
    <div id="result">結果:</div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("checkButton").onclick = function(){
    var text = document.getElementById("targetText").value; ―1
    var str = document.getElementById("searchText").value; ―2
    var ptr = text.indexOf(str,0); ―3
    if (ptr < 0) { ―4
      document.getElementById("result").innerHTML = "見つかりませんでした";
    }else{
      document.getElementById("result").innerHTML = 
      (ptr+1)+"文字目で見つかりました";
    }
  }
}
```

1 …検索対象の文字列を読み出す

2 …検索文字列を読み出す

3 …文字列の最初から検索を行う

4 …「indexOf()」の戻り値が「−1」の場合は文字列がなかったことになり、「0」以上の場合は文字列が見つかったことになるので、それぞれ処理を行う

OnePoint 文字列を検索するには「indexOf()」「lastIndexOf()」を使う

文字列を検索するには、「indexOf()」または「lastIndexOf()」を使います。「indexOf()」は、文字列の先頭から検索して見つかった位置を返します。「lastIndexOf()」は、文字列の末尾から検索して見つかった位置を返します。また、検索位置を指定することで文字列の途中から検索することもできます。文字列が見つからなかった場合は「−1」を返します。

なお、より高度な検索を行いたい場合には、正規表現が利用できる「match()」を利用します(274ページ参照)。

条件にマッチする文字列や数値があるか調べる

ここでは、正規表現を使って、「ru」で始まる文字列を検索する方法について解説します。

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; 
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css" 
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <form action="./check.cgi" method="get" name="mainForm">
      言語名:<input type="text" name="langName" 
      id="langName" value="java perl ruby Jython">
      <input type="button" id="checkButton" 
      value="ruで始まる文字列にマッチする単語を調べる">
    </form>
    <div id="result">結果:</div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("checkButton").onclick = function(){
    var text = document.getElementById("langName").value; ―1
    var resultArray = text.match(/ru+/gi); ―2
    document.getElementById("result").innerHTML = ↩
    resultArray.length+"個見つかりました。";
  }
}
```

1 …検索対象の文字列を読み出す

2 …「match()」を使って「ru」の文字で始まる文字列を検索する

OnePoint 正規表現を使って検索するには「match()」を使う

正規表現を使って検索するには、「match()」を使います。正規表現を利用すると、複雑な条件にマッチする文字列を検索することができます。「match(/ab/)」とすると「ab」の文字が含まれる文字列にマッチします。ただし、複数マッチする場合でも最初の1つにマッチした時点で検索が終わります。文字列全体を検索する場合には、「g」オプションを付加して「match(/ab/g)」のように指定します。「ab」だけでなく「Ab」や「aB」など大文字小文字に関係なくマッチさせたい場合には、「i」オプションを付加して「match(/ab/i)」のように指定します。

JavaScriptで利用できる正規表現のパターンは、下表のようになります。

パターン	内　容
.	任意文字にマッチ(改行は除く)
[文字]	[]内にある任意文字にマッチ
[^文字]	[]内以外の任意文字にマッチ
文字|文字	|で区切られた任意文字にマッチ
文字*	0個以上の出現でマッチ
文字+	1個以上の出現でマッチ
文字?	0または1個の出現でマッチ
文字{値}	値の指定個数でマッチ
文字{値,}	値の指定個数以上でマッチ
文字{値1,値2}	値1～値2の指定範囲でマッチ
^文字	先頭の文字にマッチ
文字$	末尾にマッチ
文字¥b	空白で区切られた単語にマッチ
文字¥B	空白で区切られていない単語にマッチ
¥c文字	コントロールコードの文字にマッチ
¥d	数値にマッチ([0-9]と同じ)
¥D	数値以外の文字にマッチ ([^0-9]と同じ)
¥f	フォームフィードの文字にマッチ
¥n	改行文字にマッチ

パターン	内　容
¥r	復帰文字にマッチ
¥s	1文字の区切り文字にマッチ ([¥f¥n¥r¥t¥v]と同じ)
¥S	区切り文字以外の1文字にマッチ ([¥f¥n¥r¥t¥v]と同じ)
¥t	タブ文字にマッチ
¥v	垂直タブ文字にマッチ
¥w	英数文字にマッチ ([A-Za-z0-9_]と同じ)
¥W	英数文字以外の文字にマッチ ([^A-Za-z0-9_]と同じ)
¥★	★(数値)番目の文字列(...)にマッチ
¥o値	8進数の値にマッチ
¥x値	16進数の値にマッチ
¥.	「.」の文字
¥-	「-」の文字
¥¥	「¥」の文字
¥/	「/」の文字
()	グループ化(マッチデータは RegExp.$1～9で取り出せる)

119 文字列をエンコード/デコードする

ここでは、文字列をエンコード/デコードする方法について解説します。

ボタンをクリックすると…

文字列のエンコード/デコード結果が表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
```

```
  <body>
    <h1>文字列のエンコード/デコードのチェック</h1>
    <form action="./check.cgi" method="get" name="mainForm">
      文字:<input type="text" name="convCode" id="convCode" 
      value="#mz-80K2E%21%21"><br>
      <input type="button" id="checkButton1" value="エンコードする"><br>
      <input type="button" id="checkButton2" value="デコードする">
    </form>
    <div id="result">結果:</div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("checkButton1").onclick = function(){
    var text = document.getElementById("convCode").value;
    var encText = encodeURIComponent(text); ―1
    document.getElementById("result").innerHTML = 
    "エンコード結果:"+encText;
  }
  document.getElementById("checkButton2").onclick = function(){
    var text = document.getElementById("convCode").value;
    var encText = decodeURIComponent(text); ―2
    document.getElementById("result").innerHTML = "デコード結果:"+encText;
  }
}
```

1 …文字列をエンコードする

2 …文字列をデコードする

OnePoint エンコード/デコードには「encodeURIComponent()」「decodeURIComponent()」を使う

CGIなどに日本語や記号などを含むデータをURL文字列として送信する場合、エンコード処理を行う必要があります。JavaScriptには、エンコード／デコード処理を行うメソッドがいくつか用意されています(下表参照)。ただし、古いブラウザでは、「escape()」「unescape()」しか利用できない場合があります。エンコード処理を行うメソッドは、それぞれどの文字列をエンコードするかが異なっています。たとえば、「encodeURIComponent()」では「;」「/」「_」「:」「@」「&」「=」「+」「$」「,」の文字までエンコードされるので、これらの文字をエンコードしたくない場合は、「encodeURI()」を使います。

エンコード	デコード
escape()	unescape()
encodeURI()	decodeURI()
encodeURIComponent()	decodeURIComponent()

SECTION 120

1文字ずつページ上に文字を表示してクリックで一括表示する

ここでは、1文字ずつページ上に文字を表示しつつ、表示領域をクリックしたときには一括表示する方法について解説します。

ボタンをクリックすると…

文字が1文字ずつ表示され、文字表示領域をクリックするとすべての文字が一括して表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>管理人からのメッセージ</h1>
    <form action="./check.cgi" method="get" name="mainForm">
      <input type="button" id="dispButton" value="表示">
    </form>
    <div id="result">---</div> ―1
  </body>
</html>
```

1 …文字列の表示領域を指定する

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("dispButton").onclick = function(){
    message.start("result",
    "本日はようこそいらっしゃいました。どうぞごゆっくりとしていってください。");
  }
}
var message = {
  oneChar : function(){
    document.getElementById(this.outID).innerHTML =
      this.text.substr(0, this.count);                    1
    this.count++;
    if (this.count <= this.text.length)
    setTimeout("message.oneChar()", 100);                2
  },
  allChar : function(){
    document.getElementById(message.outID).innerHTML =
    message.text;                                         3
    message.count = message.text.length;
  },
  start : function(oID, str){
    this.text = str;
    this.count = 0;
    this.outID = oID;
    document.getElementById(this.outID).onclick = message.allChar;  4
    setTimeout("message.oneChar()", 100);                5
  }
}
```

1 …文字列をカウンタで示される数だけ表示する

2 …文字列が全部表示されていない場合のみタイマーを設定し表示処理を行う

3 …すべての文字列を表示する処理

4 …クリック時にすべて表示する処理を呼び出す

5 …1文字ずつ表示する処理を0.1秒後に呼び出す

OnePoint 「setTimeout()」を使って定期的に文字を表示する

1文字ずつページに文字列を表示するには、タイマーを利用します。JavaScriptのタイマーには「setTimeout()」と「setInterval()」の2種類がありますが、処理に終わりがある場合には「setTimeout()」を使う方が安全です。文字列の表示にはカウンタを用意し、文字列の表示位置を示すようにします。タイマーで定期的にカウンタを増やし、文字列を表示していきます。カウンタの値が文字列よりも少ない場合はタイマーを設定して繰り返し表示処理を行います。

クリック時にすべての文字列を表示するには、表示領域にクリックイベント(onclick)を指定します。イベントが発生したらすべての文字列を表示し、カウンタを文字列長と同じにします。

SECTION 121 ページが読み込まれたら更新情報をページ内に流し込む

ここでは、ページが読み込まれたタイミングで、指定した領域にテキストファイルのデータを流し込む方法について解説します。

ページが読み込まれると、それぞれの領域に自動的にテキスト/HTMLファイルが流し込まれる

SOURCE CODE　HTML ▶ index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
```

```
    <meta http-equiv="content-type" content="text/html; ⤶
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css" ⤶
    media="all">
    <script type="text/javascript" src="prototype.js"></script>
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>更新情報</h1>
    <div id="header">ヘッダーを読み込み中....</div>
    <div id="info">最新情報を読み込み中....</div>        1
    <div id="message">本文を読み込み中....</div>
  </body>
</html>
```

1 …ファイルを読み込む領域を指定する(ID名も指定しておく)

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  new Ajax.Updater("header", "header.txt"); —1
  new Ajax.Updater("info", "info.txt"); —2
  new Ajax.Updater("message", "message.txt"); —3
}
```

1 …ID名「header」のタグにheader.txtファイルを流し込んで表示する
2 …ID名「info」のタグにinfo.txtファイルを流し込んで表示する
3 …ID名「message」のタグにmessage.txtファイルを流し込んで表示する

OnePoint 非同期でページ内にテキスト/HTMLファイルを表示するには「new Ajax.Updater()」を使う

非同期でページ内にテキスト/HTMLファイルを表示するには、prototype.jsライブラリの「new Ajax.Updater()」を使うと簡単です。「new Ajax.Updater()」の最初のパラメータに読み込むタグのID名を指定します。2番目のパラメータには、読み込むファイルのURLを指定します。ただし、セキュリティの都合上、同一ドメイン、同一サーバー上にあるファイルしか読み込むことはできません。

OnePoint Internet Explorer7では「XMLHttpRequest」が使える

Internet Explorer6まではActive Xを利用して非同期通信を行っていましたが、Internet Explorer7からは他のブラウザと同様に「XMLHttpRequest」を使うことができます。ただし、「XMLHttpRequest」はブラウザによって利用できるイベントやプロパティが異なるので注意が必要です。

Break Time

文字列の連結をやめて「join()」を使って実行速度を上げる

JavaScriptでは、「+」記号を使って手軽に文字列の連結を行うことができます。ところが、Internet Explorerでは、この文字列の連結が非常に低速な処理となり、実行速度を低下させてしまうことがあります。このような場合には連結する文字列を一度まとめて配列に入れておき、最後に「join()」で連結するようにします。これで、だいたい100倍ほど処理速度が向上します。ただし、他のブラウザではあまり効果がありません。

```
window.onload = function(){
  txt = "";
  s = new Date();
  for (var i=0; i<100000; i++){
    txt += "Sample text, ";
  }
  e = new Date();
  tm1 = (e-s) / 1000;

  txt2 = [];
  s = new Date();
  for (var i=0; i<100000; i++){
    txt2[i] = "Sample text, ";
  }
  txtX = txt2.join("");
  e = new Date();
  tm2 = (e-s) / 1000;

  document.getElementById("time").innerHTML = tm1+"<br>"+tm2;
}
```

JavaScript Sample - Windows Internet Explorer
G:¥章末コラム¥第!
Live Search
J...
Jav...
サンプル
156.745
0.157

※Internet Explorerでこのサンプルを実行すると処理に時間がかかるので注意してください。

Internet Explorerでは、「join()」を使った方が実行速度が速くなる

CHAPTER - 10

日付/時刻

SECTION

122 現在の日付を表示する

ここでは、ページを表示した時点の日付を、西暦の年月日と曜日で表示する方法について解説します。

JavaScript Sample - Windows Internet Explorer

今日は2007年3月31日(土)です。

ページが読み込まれた時点での日付が表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <p>今日は<span id="currentDate"></span>です。</p> ―1
  </body>
</html>
```

1 …日付を表示する領域を用意する

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  var dateObj = new Date(); ―1
  var y = dateObj.getFullYear(); ―2
  var m = dateObj.getMonth() + 1; ―3
```

```
    var d = dateObj.getDate(); —4
    var yb = "日月火水木金土".charAt(dateObj.getDay()); —5
    document.getElementById("currentDate").innerHTML =
    y+"年"+m+"月"+d+"日("+yb+")";                        ]6
}
```

1 …日付オブジェクトを生成する
2 …「getFullYear()」で西暦年数(4桁)を読み出す
3 …「getMonth()」で月を読み出して「1」を加算する(「getMonth()」で取得できる値は実際の月より1少なくなるため)
4 …「getDate()」で日にちを読み出す
5 …「getDay()」で曜日を0〜6までの数値で取得し、それをキーにして「charAt()」で曜日の文字列から抜き出す
6 …日付を文字列として連結しページ上に表示する

OnePoint 「Date」オブジェクトの各種メソッドを使って日付情報を読み出す

現在の日付を読み出すには「Date」オブジェクトを生成します。「Date」オブジェクトは生成された時点での日付情報であり、リアルタイムに変化することはありません。リアルタイムに日付情報を取得する場合には、毎回、「Date」オブジェクトを生成する必要があります。

生成された「Date」オブジェクトから日付情報を読み出すには、下表の各種メソッドを利用します。

メソッド	内　容
getFullYear()	西暦年数(4桁)を取得する
getMonth()	月を取得する
getDate()	日にちを取得する
getDay()	曜日を取得する

注意しないといけないのは、月を示す「getMonth()」です。これは、実際の月よりも「1」少ない値を返すため、表示する際には「1」を加算する必要があります。

また、曜日を示す「getDay()」は、曜日に応じて0〜6までの数値を返します。これは下表のように曜日と対応しています。曜日を表示する場合には、あらかじめ配列に曜日を示す文字列を入れておくか、サンプルのように文字列内から抜き出すようにします。

数　値	曜　日
0	日
1	月
2	火
3	水
4	木
5	金
6	土

SECTION

123 現在の時刻を表示する

ここでは、ページを表示した時点の時刻を、時分秒で表示する方法について解説します。

JavaScript Sample - Windows Internet Explorer

Welcome!

このページにアクセスした時刻は11時55分1秒です。

ページが読み込まれた時点での時刻が表示される

SOURCE CODE　HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>Welcome!</h1>
    <p>このページにアクセスした時刻は<span id="currentTime"></span>です。</p> ―1
  </body>
</html>
```

1 …時刻を表示する領域を用意する

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  var dateObj = new Date(); ─1
  var h = dateObj.getHours(); ─2
  var m = dateObj.getMinutes(); ─3
  var s = dateObj.getSeconds(); ─4
  document.getElementById("currentTime").innerHTML = ⤶ ]─5
  h+"時"+m+"分"+s+"秒";
}
```

1 …日付オブジェクトを生成する
2 …「getHours()」で時を読み出す
3 …「getMinutes()」で分を読み出す
4 …「getSeconds()」で秒を読み出す
5 …得られた時刻を文字列として連結しページ上に表示する

OnePoint 「Date」オブジェクトの各種メソッドを使って時刻情報を読み出す

現在の時刻を読み出すには、「Date」オブジェクトを生成します。「Date」オブジェクトは生成された時点での時刻の情報であり、リアルタイムに変化することはありません。リアルタイムに時刻情報を取得する場合には、毎回、「Date」オブジェクトを生成する必要があります。

生成された「Date」オブジェクトから時刻の情報を読み出すには、下表の各種メソッドを利用します。

メソッド	内 容
getHours()	時を取得する
getMinutes()	分を取得する
getSeconds()	秒を取得する
getMilliseconds()	ミリ秒を取得する

Column 「Date」オブジェクトに設定した日付や時刻

「Date」オブジェクトは日付や時刻を読み出すだけでなく、設定することができます。なお、「Date」オブジェクトに設定した日付や時刻はシステム(コンピュータ)の日付/時刻には影響を与えません。あくまでも「Date」オブジェクト内の日付/時刻の設定になります。

メソッド	内 容
setFullYear	西暦年を設定する
setMonth	月を設定する
setDate	日にちを設定する
setTime	時間を設定する
setHours	時を設定する
setMinutes	分を設定する
setSeconds	秒を設定する
setMilliseconds	ミリ秒を設定する

124 n日後を求める

ここでは、テキストフィールドに入力した日数を現在の日付に加算する方法について解説します。

日数：3
n日後を求める
結果：

日数を入力してボタンをクリックすると…

日数：3
n日後を求める
3日後は「2007年4月3日(火)」です

現時点での日付に入力した日数を加算して表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <form action="./check.cgi" method="get" name="mainForm">
```

```
        日数:<input type="text" name="dateCount"
        id="dateCount" value="3">
        <input type="button" id="checkButton" value="n日後を求める">
      </form>
      <div id="result">結果:</div>
   </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("checkButton").onclick = function(){
    var dc = document.getElementById("dateCount").value; ―1
    dc = parseInt(dc);
    if (isNaN(dc)) { alert("入力エラー"); return; }
    var dateObj = new Date();
    var today = dateObj.getTime(); ―2
    var msec = dc * (24 * 60 * 60 * 1000); ―3
    var newday = new Date(today+msec); ―4
    var y = newday.getFullYear();
    var m = newday.getMonth() + 1;
    var d = newday.getDate();
    var yb = "日月火水木金土".charAt(newday.getDay());
    document.getElementById("result").innerHTML =
    dc+"日後は「"+y+"年"+m+"月"+d+"日("+yb+")」です";
  }
}
```

1 …入力された日数を取得する

2 …現在の日付の1970年1月1日からの経過秒数(ミリ秒)を取得する

3 …取得した日数にミリ秒を乗算する

4 …現在の日付の経過秒数と入力された日数を秒数(ミリ秒)に変換した値を加算して「Date()」のパラメータとして指定する(これにより指定された経過秒数の「Date」オブジェクトが生成される)

OnePoint 「Date」オブジェクトの引数に経過秒数を指定する

指定した経過日数を求めるには現在の日付を示す経過秒数(1970年1月1日からの経過ミリ秒)と、求める日数を示すミリ秒を加算/減算します。この値を「Date()」パラメータとして指定し、新たなオブジェクトとして生成します。あとは、「getYear()」や「getMonth()」などで日付情報を読み出すことができます。

「Date」オブジェクトは内部では1970年1月1日からの経過ミリ秒で計算されるため、このような方法で手軽に日数を計算することができます。

SECTION

125 実行時間を計測する

ここでは、プログラム実行時の処理にかかった時間を計測する方法について解説します。

JavaScript Sample - Windows Internet Explorer
実行時間の計測
回数: 10000
実行時間を計測
結果:

回数を入れてボタンを
クリックすると…

JavaScript Sample - Windows Internet Explorer
実行時間の計測
回数: 10000
実行時間を計測
6547ミリ秒かかりました

処理にかかった時間が
表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
```

```
    <h1>実行時間の計測</h1>
    <form action="./check.cgi" method="get" name="mainForm">
      回数:<input type="text" name="repeatCount" id="repeatCount"
      value="10000">
      <input type="button" id="checkButton" value="実行時間を計測">
    </form>
    <div id="result">結果:</div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("checkButton").onclick = function(){
    var count =
    parseInt(document.getElementById("repeatCount").value);
    var str = "";
    var startTime = new Date();  ―1
    for (var i=0; i<count; i++){
      str += "JavaScript Sample Text, "+Math.pow(10,20)/2+Math.PI;
    }                                                    ―2
    var endTime = new Date();  ―3
    var msec = endTime - startTime;  ―4
    document.getElementById("result").innerHTML = msec+"ミリ秒かかりました";
  }
}
```

1 …開始時の時間を求める

2 …計測したい処理を記述する

3 …終了時点での時間を求める

4 …終了時刻から開始時刻を減算してかかった時間を求める

OnePoint 開始時と終了時に生成した「Date」オブジェクトの差分を求める

処理にかかる時間を計測するには、計測開始時と終了時に「Date」オブジェクトを生成し、差分を求めます。生成された「Date」オブジェクトは1970年1月1日からの経過秒数(ミリ秒)なので、差分を取れば処理にかかった時間を求めることができます。

```
startTime = new Date();
  計測したい処理
endTime = new Date();
msec = endTime - startTime;

msec : 処理にかかった時間(ミリ秒)
```

SECTION

126 リアルタイムに時刻をテキストで表示する

ここでは、テキストでリアルタイムに時刻を表示する方法について解説します。

現在の時刻:11時56分43秒

現在の時刻:11時56分51秒

ページが表示されると現在の時刻がリアルタイムに表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; ⤵
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css" ⤵
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <p>現在の時刻:<span id="currentTime"></span></p>
  </body>
</html>
```

SOURCE CODE　JavaScript main.js

```
window.onload = function(){
  myClock.time(); —1
  setInterval("myClock.time()", 1000); —2
}
var myClock = {
  time : function(){
    var dateObj = new Date();
    var h = dateObj.getHours();
    var m = dateObj.getMinutes();
    var s = dateObj.getSeconds();
    document.getElementById("currentTime").innerHTML = ⤶
    h+"時"+m+"分"+s+"秒";
  }
}
```

1 …現在の時間を表示する(この処理がないと1秒ほど時刻が表示されない)

2 …1秒ごとに時間を表示する「myClock.time()」を呼び出す

3 …「Date」オブジェクトを生成して時間を求め、求めた時間をページ内に表示する

OnePoint リアルタイムに時刻を表示するには「setInterval()」を使う

リアルタイムに時刻を表示するには、「setInterval()」を使って時刻を表示する処理を呼び出します。「setInterval()」の最初のパラメータに定期的に行う処理を記述し、2番目のパラメータに定期的に呼び出す時間をミリ秒で指定します。ミリ秒なので1秒ごとに呼び出すのであれば、「1000」を指定します。

Column 先頭に0を付加して表示する

時間が0〜9までの場合に、先頭に0を付加して2桁で表示させたい場合があります。これは、次のように文字として0を付加した後に「substring()」を使って末尾から2文字分を抜き出します。

```
var myClock = {
  time : function(){
    var dateObj = new Date();
    var h = "0"+dateObj.getHours();
    var m = "0"+dateObj.getMinutes();
    var s = "0" + dateObj.getSeconds();
    h = h.substring(h.length-2, h.length);
    m = m.substring(m.length-2, m.length);
    s = s.substring(s.length-2, s.length);
    document.getElementById("currentTime").innerHTML = ⤶
    h+"時"+m+"分"+s+"秒";
  }
}
```

SECTION 127

リアルタイムに時刻をデジタル時計風の画像で表示する

ここでは、デジタル時計のような画像を使ってリアルタイムに時刻を表示する方法について解説します。

ページが表示されると、現在の時刻がリアルタイムに画像で表示される

JavaScript Sample - Windows Internet Explorer

現在の時刻は

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>現在の時刻は</h1>
    <div id="iTime"> ―1
      <img src="images/0.gif" width="20" height="30"
      ><img src="images/1.gif" width="20" height="30"
      ><img src="images/colon.gif" width="10" height="30"
      ><img src="images/2.gif" width="20" height="30"
      ><img src="images/3.gif" width="20" height="30"
      ><img src="images/colon.gif" width="10" height="30"
      ><img src="images/4.gif" width="20" height="30"
      ><img src="images/5.gif" width="20" height="30">
    </div>
  </body>
</html>
```

1 …画像を表示する領域を<div>タグで指定してID名を付けておく

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  myClock.timeImage(); ─1
  setInterval("myClock.timeImage()", 1000); ─2
}
var myClock = {
  timeImage : function(){
    var dateObj = new Date();
    var h = "0"+dateObj.getHours();
    var m = "0"+dateObj.getMinutes();
    var s = "0"+dateObj.getSeconds();
    h = h.substring(h.length-2, h.length);
    m = m.substring(m.length-2, m.length);
    s = s.substring(s.length-2, s.length);
    var imageTag =
    document.getElementById("iTime").getElementsByTagName("img");
    imageTag[0].src = "images/"+h.charAt(0)+".gif";
    imageTag[1].src = "images/"+h.charAt(1)+".gif";
    imageTag[3].src = "images/"+m.charAt(0)+".gif";
    imageTag[4].src = "images/"+m.charAt(1)+".gif";
    imageTag[6].src = "images/"+s.charAt(0)+".gif";
    imageTag[7].src = "images/"+s.charAt(1)+".gif";
  }
}
```

1 …現在の時間を表示する(この処理がないと1秒ほど時刻が表示されない)

2 …1秒ごとに時間を表示する「myClock.timeImage()」を呼び出す

3 …「Date」オブジェクトを生成して時間を求め、値が1桁の場合は2桁にして文字列に変換する

4 …表示領域にある<img>タグを「getElementsByTagName()」で求める

5 …求めた<img>タグに表示する時刻の画像URLを指定する

OnePoint 画像で時刻を表すには<img>タグの「src」プロパティに時刻を示す値に対応した画像を設定する

リアルタイムに時刻を画像で表示するには、「setInterval()」を使って時刻を表示する処理を呼び出します。呼び出された関数側では、「Date」オブジェクトを生成して時刻を求めます。この時刻の数値に対応した0.gif〜9.gifの画像を用意しておきます。時刻の値に応じて画像のURLを「src」プロパティに設定します。

画像を操作する場合、通常は「id」属性または「name」属性で名前を指定しますが、サンプルのように表示領域内にある<img>タグを求めるようにすると、「id」属性や「name」属性で名前を指定する必要がなくなります。これによりIDや名前を管理する必要がなくなり、トラブルも防ぐことができます。

128 リアルタイムに時刻をアナログ時計で表示する

ここでは、画像を使って、アナログ時計でリアルタイムに時刻を表示する方法について解説します。

※このサンプルはFirefox1.5以降、Opera9以降、Safari2以降でないと動作しません。

現在の時刻がリアルタイムにアナログ時計で表示される

JavaScript Sample - Mozilla Firefox
ファイル(F) 編集(E) 表示(V) 履歴(S) ブックマーク(B) ツール(T) ヘルプ(H)
file:///C:/JavaScr
Firefox を使ってみよう 最新ニュース
アナログ時計
K.FuRuhata

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; ↩
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css" ↩
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>アナログ時計</h1>
    <canvas id="clock" width="128" height="128"></canvas> ―1
  </body>
</html>
```

1 …アナログ時計を表示する<canvas>タグを用意してID名を指定する

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  canvasObj = document.getElementById("clock"); ┐
  conObj = canvasObj.getContext("2d");          ┘1
  setInterval("imageClock.timeImage()", 1000); ―2
}
var imageClock = {
  timeImage : function(){
```

```
    var dateObj = new Date();
    var H = dateObj.getHours() % 12;
    var M = dateObj.getMinutes();
    var S = dateObj.getSeconds();
    conObj.drawImage(imgBaseObj,0,0); ―3
    this.draw(H*30, imgHObj);   ┐
    this.draw(M*6, imgMObj);    ├4
    this.draw(S*6, imgSObj);    ┘
    conObj.drawImage(imgCenterObj,56,56); ―5
  },
  draw : function(n, img){
    conObj.save();                              ┐
    conObj.translate(64,64);                    │
    conObj.rotate((180+n) * Math.PI / 180);     ├6
    conObj.drawImage(img,0,0);                  │
    conObj.restore();                           ┘
  }
}
imgBaseObj = new Image(128,128);
imgBaseObj.src = "images/clock.png";
imgCenterObj = new Image(16,17);
imgCenterObj.src = "images/pin.png";
imgHObj = new Image(3,64);
imgHObj.src = "images/h.png";
imgMObj = new Image(3,64);
imgMObj.src = "images/m.png";
imgSObj = new Image(3,64);
imgSObj.src = "images/s.png";
```

1 …<canvas>タグで描画領域を扱えるようにする

2 …1秒ごとに時間を表示する「imageClock.timeImage()」を呼び出す

3 …アナログ時計のベース画像を描画する

4 …時分秒の針を描画する

5 …時計の中央に画像を表示する

6 …針の画像を回転させて描画する

OnePoint アナログ時計は画像を重ねて時計を表現する

Firefox1.5以降、Safari2以降、Opera9以降では<canvas>タグにより自由にグラフィックの描画を行うことができます。<canvas>タグには「width」属性と「height」属性を指定して描画サイズを指定します。描画サイズを指定しないと、ブラウザによっては描画結果が異なることがあります。用意したキャンバスに描画を行うには、「getContext("2d")」として平面に描画するためのオブジェクトを取得します。このオブジェクトに対して「drawImage()」で指定したURLの画像を描画します。アナログ時計の場合、時計の背面板、針などの画像をPNG形式で個別に用意し、「drawImage()」で順番に描画していきます。

針の表示位置(原点)は、「translate()」で指定します。針の回転を表現するには、「rotate()」を使います。この「rotate()」のパラメータは回転角度で、ラジアンで指定します。「rotate()」を指定すると、それ以後に描画される図形や画像は、すべてて指定した角度だけ回転した状態で描画されます。

移動や回転を行うと描画角度や原点がずれてしまうため、「save()」を使って描画前の描画面に関する情報を保存します。保存した情報を元に戻すには、「restore()」を使います。

Break Time

「Date」オブジェクトの扱い

JavaScriptでは、日付を扱う「Date」オブジェクトが用意されています。この「Date」オブジェクトは、指定された日付の情報を生成します。この「Date」オブジェクトですが、内部では1970年1月1日午前0時から指定した時間までのミリ秒となっています。このため、「Date」オブジェクト同士であれば、次のサンプルのように加算や減算を行うことができます。

また、「setMonth()」や「setDate()」などでは、「setDate(60)」のように日にちの範囲を超えて指定した場合でもミリ秒で計算されるため、正しく日付処理が行われます。

「new Date()」として日付オブジェクトを生成する際にパラメータを指定することができますが、このパラメータは日付文字列だけでなく、1970年1月1日からの経過ミリ秒も指定することができます。これによりミリ秒を基準としてn日後やnヶ月後などの日付を手軽に求めることができます。

```
window.onload = function(){
  tm1 = new Date();
  alert("少し待ってからOKボタンをクリック");
  tm2 = new Date();
  alert(tm2-tm1+"ミリ秒後にクリックされました");
}
```

Windows Internet Explorer

少し待ってからOKボタンをクリック

OK

Windows Internet Explorer

15723ミリ秒後にクリックされました

OK

アラートが表示されてからボタンをクリックするまでの時間が表示される

CHAPTER - 11

その他

SECTION 129 タブ区切り形式のデータを読み込んで表示する

ここでは、タブ区切り形式のデータを読み込んで、表形式で表示する方法について解説します。

JavaScript Sample - Windows Internet Explorer

http://192.168.1.22/11_etc/11

おすすめパソコンの価格一覧

品名	価格	注釈
ビジネス向けノートPC	87,800円	ビジネスユースに最適
プライベート向けノートPC	129,800円	プライベートでもストレスなく使えます
高性能デスクトップPC	199,800円	ゲーム用途に最適
サーバー	59,800円	低価格のサーバー

ページが表示されると、タブ区切りのデータがテーブルとして表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="prototype.js"></script> ―1
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>おすすめパソコンの価格一覧</h1>
    <div id="tableData">データを読み込み中....</div>
  </body>
</html>
```

1 …prototype.jsライブラリを読み込む

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  var msec = (new Date()).getTime(); 1
  new Ajax.Request("data.txt", { 2
    method: "get",
    parameters: "cache="+msec,
    onSuccess:function(httpObj){
      var text = httpObj.responseText; 3
      var LF = String.fromCharCode(10); 4
      var TAB = String.fromCharCode(9); 5
      var tabText = text.split(LF); 6
      var tbl = "<table border='1'>";
      for (var i=0; i<tabText.length; i++){
        var cText = tabText[i].split(TAB); 8
        tbl += "<tr>";
        for (var j=0; j<cText.length; j++){
          tbl += "<td>"+cText[j]+"</td>";          7
        }
        tbl +="</tr>";
      }
      tbl += "</table>";
      $("tableData").innerHTML = tbl;
    },
    onFailure:function(httpObj){
      $("tableData").innerHTML = "エラーで読み込めませんでした";
    }
  });
}
```

1 …ローカルキャッシュさせないために現在のミリ秒とURLを組み合わせる
2 …読み込むファイル(data.txt)を指定する
3 …非同期で読み込まれたタブ区切りテキストを読み込む
4 …改行コードを設定する
5 …タブコードを設定する
6 …改行コードで分割する
7 …データの数だけテーブルとして生成する
8 …タブコードで分割する

OnePoint prototype.jsライブラリを利用して非同期で読み込ませタブコードで分割する

タブ区切り形式のテキストデータを読み込ませ表示するには、prototype.jsライブラリを利用すると簡単です。「Ajax.Request()」で読み込むタブ区切りテキストのURLを指定します。データが読み込まれたら、改行コードとタブコードで分割します。分割された数だけ繰り返し、テーブルとして表示します。

130 CSV形式のデータを読み込んで表示する

ここでは、CSV形式のデータを読み込んで、表形式で表示する方法について解説します。

JavaScript Sample - Windows Internet Explorer

コーヒー豆の価格一覧

品名	価格	注釈
モカ	800円	実のような香りとコクが特徴
キリマンジャロ	800円	独特の酸味が特徴
マンデリン	1000円	苦みとコク、芳醇な香りが特徴
ブルーマウンテン	1500円	最高級のコーヒー豆

ページが表示されると、CSV形式のデータがテーブルとして表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="prototype.js"></script> ―1
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>コーヒー豆の価格一覧</h1>
    <div id="tableData">データを読み込み中....</div>
  </body>
</html>
```

1 …prototype.jsライブラリを読み込む

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  var msec = (new Date()).getTime(); ―1
  new Ajax.Request("data.csv", { ―2
    method: "get",
    parameters: "cache="+msec,
    onSuccess:function(httpObj){
      var text = httpObj.responseText; ―3
      var LF = String.fromCharCode(10); ―4
      var tabText = text.split(LF); ―5
      var tbl = "<table border='1'>";
      for (var i=0; i<tabText.length; i++){
        var cText = tabText[i].split(","); ―7
        tbl += "<tr>";
        for (var j=0; j<cText.length; j++){
          tbl += "<td>"+cText[j]+"</td>";
        }
        tbl +="</tr>";
      }
      tbl += "</table>";
      $("tableData").innerHTML = tbl;
    },
    onFailure:function(httpObj){
      $("tableData").innerHTML = "エラーで読み込めませんでした";
    }
  });
}
```

1 …ローカルキャッシュさせないために現在のミリ秒とURLを組み合わせる

2 …読み込むファイル(data.csv)を指定する

3 …非同期で読み込まれたCSV形式のテキストを読み込む

4 …改行コードを設定する

5 …改行コードで分割する

6 …データの数だけテーブルとして生成する

7 …「,」(カンマ)で分割する

OnePoint prototype.jsライブラリを利用して非同期で読み込ませ「,」で分割する

CSV形式のテキストデータを読み込ませ表示するには、prototype.jsライブラリを利用すると簡単です。「Ajax.Request()」で読み込むCSV形式のテキストのURLを指定します。データが読み込まれたら、改行コードと「,」(カンマ)で分割します。分割された数だけ繰り返し、テーブルとして表示します。

サンプルで扱っているCSVデータは、各データには「,」(カンマ)が含まれず、なおかつ「"」(ダブルクォーテーション)で囲まれていないデータを使用しています。Excelやデータベースから出力されるデータの場合、「"」で囲まれているので表示する際には、「"」の削除などの処理が必要になります。

SECTION 131

XML形式のデータを読み込んで表示する

ここでは、XML形式のデータを読み込んで、表形式で表示する方法について解説します。

JavaScript Sample - Windows Internet Explorer

http://192.168.1.22/11_etc/11

Windows Vistaの価格一覧

Home Basic	27090円	最低限のパッケージです
Home Premium	31290円	家庭用です
Business	39690円	ビジネス用途に
Ultimate	51240円	フルスペックのVistaです

ページが表示されると、XML形式のデータがテーブルとして表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; 
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css" 
    media="all">
    <script type="text/javascript" src="prototype.js"></script> ―1
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>Windows Vistaの価格一覧</h1>
    <div id="tableData">データを読み込み中....</div>
  </body>
</html>
```

1 …prototype.jsライブラリを読み込む

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  var msec = (new Date()).getTime(); ―1
  new Ajax.Request("data.xml", { ―2
    method: "get",
    parameters: "cache="+msec,
    onSuccess:function(httpObj){
```

```
      var xmlData = httpObj.responseXML; ─3
      var winTag = xmlData.getElementsByTagName("windows"); ─4
      var tbl = "<table border='1'>";
      for (var i=0; i<winTag.length; i++){ ─5
        tbl += "<tr>";
        var typ = winTag[i].getElementsByTagName("type")[0];
        var price = winTag[i].getElementsByTagName("price")[0];   6
        var note = winTag[i].getElementsByTagName("note")[0];
        tbl += "<td>"+typ.firstChild.nodeValue+"</td>";
        tbl += "<td>"+price.firstChild.nodeValue+"</td>";   7
        tbl += "<td>"+note.firstChild.nodeValue+"</td>";
        tbl +="</tr>";
      }
      tbl += "</table>";
      $("tableData").innerHTML = tbl;
    },
    onFailure:function(httpObj){
      $("tableData").innerHTML = "エラーで読み込めませんでした";
    }
  });
}
```

1 …ローカルキャッシュさせないために現在のミリ秒とURLを組み合わせる

2 …読み込むファイル(data.xml)を指定する

3 …XMLデータ(DOM)を読み込む

4 …<windows>タグ情報を取得する

5 …<windows>タグの数だけ繰り返す

6 …<type><price><note>タグの一番最初のタグの情報を取得する

7 …<type><price><note>タグの一番最初の子ノードの値を取得する

SOURCE CODE XML data.xml

```
<?xml version="1.0" encoding="utf-8"?>
<list>
  <windows><type>Home Basic</type><price>27090円</price>
  <note>最低限のパッケージです</note></windows>
  <windows><type>Home Premium</type><price>31290円</price>
  <note>家庭用です</note></windows>
  <windows><type>Business</type><price>39690円</price>
  <note>ビジネス用途に</note></windows>
  <windows><type>Ultimate</type><price>51240円</price>
  <note>フルスペックのVistaです</note></windows>
</list>
```

OnePoint prototype.jsライブラリを利用してDOMとしてXMLデータを処理する

XMLデータを読み込ませるには、prototype.jsライブラリを利用すると簡単です。「Ajax.Request()」で読み込むXMLデータのURLを指定します。XMLデータの場合、responseXMLとして読み込むとDOMとなるため、各タグ、ノードを手軽に処理することができます。タグは「getElementsByTagName()」、ノード処理は「childeNodes」「firstChild」「lastChild」などDOM関連のメソッドやプロパティを利用することができます。

サンプルでは305ページのXMLデータ(data.xml)を読み込ませています。プログラムではタグを「getElementsByTagName()」で取得し、その最初の子ノードの値を「nodeValue」で読み出しています。読み出したデータをテーブルとしてページ上に表示します。

Column ブラウザによってDOMツリーは異なっている

DOM(Document Object Model:ドキュメントオブジェクトモデル)が扱えるブラウザであれば、互換性がありそうですが、実際にはノードの解釈がブラウザごとに異なっています。一番多いのがタグとタグの間の空白や改行などのノードの解釈です。

Internet Explorerでは、タグとタグの間の空白ノードは無視され、存在しないことになります。これに対して他のブラウザでは、タグとタグの間のノードはテキストノードとして存在しています。たとえば、次のサンプルを実行すると、Internet Explorerでは「1」「3」の順番で表示されます(1はタグ、3はテキストノードを示す)。他のブラウザでは「3」「1」の順番で表示されます。

また、Safari2ではコメントノードは最初から存在しないことになり処理することはできません。このようなノードの解釈や処理がブラウザによって異なるという点には注意が必要です。これはAjaxを使ってXMLデータを処理する場合にもいえます。

◉HTMLファイル

```
<div id="contents">
<span>Sample text</span>
</div>
```

◉JavaScriptファイル

```
window.onload = function(){
  divObj = document.getElementById("contents");
  alert(divObj.firstChild.nodeType);
  alert(divObj.childNodes[1].nodeType);
}
```

Internet Explorerでは、「1」「3」の順番で表示される

その他のブラウザでは、「3」「1」の順番で表示される(画面はFireFfox)

JSON形式のデータを読み込んで表示する

ここでは、JSON形式のファイルを読み込んで、表形式で表示する方法について解説します。

JavaScript Sample - Windows Internet Explorer

空気清浄機の価格一覧

加湿機能付き空気清浄機	19,800円	加湿機能が付いたパーソナルタイプ
脱臭機能付き空気清浄機	26,390円	ニオイも取ります
家庭用空気清浄機	32,690円	吸引力がパワーアップ
業務用空気清浄機	89,800円	高性能の浄化力です

ページが表示されると、JSON形式のデータがテーブルとして表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="prototype.js"></script> ―1
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>空気清浄機の価格一覧</h1>
    <div id="tableData">データを読み込み中....</div>
  </body>
</html>
```

1 …prototype.jsライブラリを読み込む

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  var msec = (new Date()).getTime(); ―1
  new Ajax.Request("data.txt", { ―2
    method: "get",
    parameters: "cache="+msec,
    onSuccess:function(httpObj){
      var jsonData = eval(httpObj.responseText); ―3
      var tbl = "<table border='1'>";
      for (var i=0; i<jsonData.length; i++){ ―4
        tbl += "<tr>";
        tbl += "<td>"+jsonData[i].type+"</td>";   ┐
        tbl += "<td>"+jsonData[i].price+"</td>";  ├5
        tbl += "<td>"+jsonData[i].note+"</td>";   ┘
        tbl +="</tr>";
      }
      tbl += "</table>";
      $("tableData").innerHTML = tbl;
    },
    onFailure:function(httpObj){
      $("tableData").innerHTML = "エラーで読み込めませんでした";
    }
  });
}
```

1 …ローカルキャッシュさせないために現在のミリ秒とURLを組み合わせる

2 …読み込むJSONファイル(data.txt)を指定する

3 …JSON形式のデータを読み込み、「eval()」を使って正規化する

4 …JSONデータの数だけ繰り返す

5 …「type」「price」「note」の値を取得する

OnePoint 通常のテキストデータとして読み込ませ「eval()」を使って利用可能な状態にする

　JSON形式のデータを読み込ませるには、prototype.jsライブラリを利用すると簡単です。「Ajax.Request()」で読み込むJSONデータのURLを指定します。JSONはJavaScriptのデータ構造そのものなのですが、単純に読み込んだだけでは単なるテキストとなってしまい、配列/連想配列(ハッシュ)として利用することができません。このため、「eval()」を使って処理することで、配列/連想配列(ハッシュ)として使用できる状態に変換します。

　変換されたデータは通常の配列処理での指定だけでなく、「オブジェクト.プロパティ名」といった形式での指定も可能です。

Adobe Spryを使ってXMLデータを読み込んで表示する

ここでは、Adobe Spryフレームワークを使って、XMLデータを読み込んで、表形式で表示する方法について解説します。

名前	価格
リンゴ	100円
ミカン	150円
メロン	400円
イチゴ	210円
レモン	90円
サクランボ	200円

ページが表示されるとXML形式のデータが表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script src="xpath.js" type="text/javascript"></script>       ─ 1
    <script src="SpryData.js" type="text/javascript"></script>   ─ 1
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>果物の価格一覧</h1>
    <div spry:region="myDatabase"> ─ 2
      <table border="1" cellspacing="0">
        <tr><th>名前</th><th>価格</th></tr>
        <tr spry:repeat="myDatabase"> ─ 3
```

```
        <td>{myDatabase::name}</td><td>{myDatabase::price}円</td>   ← 4
      </tr>
    </table>
  </div>
</body>
</html>
```

1 …Adobe Sprγフレームワークを読み込む
2 …XMLデータを表示する領域を「spry:region」として指定する
3 …XMLデータを繰り返し処理する領域を「spry:repeat」として指定する
4 …{～}で囲まれた部分がXMLデータに置換される

SOURCE CODE JavaScript main.js

```
myDatabase = new Spry.Data.XMLDataSet("data.xml", "/list/fruits");   ← 1
```

1 …読み込むXMLデータファイルと処理基準となる「XPath」を指定する

OnePoint XMLデータを表示する部分はHTML文書内で{～}を使って指定する

Adobe SpryフレームワークでXMLデータ処理を行うには、xpath.jsとSpryData.jsファイルを読み込ませます。ページ上にXMLデータを表示する領域と、何のデータを表示するかを指定します。表示する領域の指定は、「spry:region」属性を使い、表示対象となるデータベース名を指定します。このデータベース名は、スクリプトファイルで設定した名前になります。

XMLデータ表示領域を設定したら、繰り返し表示する領域を指定します。繰り返し表示する場合には「spry:repeat」属性を使い、データベース名を指定します。実際にXMLデータを表示する部分は{～}で囲み、「{データベース名:タグ名}」のように指定します。

HTML文書での表示設定だけでは、XMLデータは表示されません。スクリプトで使用するデータベース名、XMLデータファイル名、データの基準位置を指定する「XPath」を指定します。これは「Spry.Data.XMLDataSet()」を使って指定します。最初のパラメータがXMLデータファイル名、2番目がXPathとなります。

Column Adobe SpryでXMLデータでなくプログラム内にデータを持つ場合

Adobe SpryはXMLファイルだけでなくプログラム内で配列/連想配列(ハッシュ)として定義されたデータも扱うことができます。この場合、次のようにデータを設定し処理を行います。

```
var fruitsData = [
{ fruits: "リンゴ", Color: "赤色", price: 100 },
{ fruits: "メロン", Color: "緑色", price: 400 },
{ fruits: "イチゴ", Color: "赤色", price: 210 } ];
myDatabase = new Spry.Data.DataSet();
myDatabase.data = fruitsData;
myDatabase.dataHash = fruitsData;
myDatabase.loadData();
```

134 該当するXMLデータだけを表示する

ここでは、Adobe Spryフレームワークを使って、該当するXMLデータだけを表示する方法について解説します。

JavaScript Sample - Windows Internet Explorer
http://192.168.1.22/11_etc/11
Live Search
JavaScript Sample
ページ(P)

価格一覧(赤い果物)

リンゴは赤色で100円

イチゴは赤色で210円

サクランボは赤色で200円

ページ　インターネット | 保護モード: 有効　100%

ページが表示されると、赤色に該当するデータだけが表示される

SOURCE CODE　HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script src="xpath.js" type="text/javascript"></script>    ─┐
    <script src="SpryData.js" type="text/javascript"></script> ─┘1
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>価格一覧(赤い果物)</h1>
    <div spry:region="myDatabase"> ─2
      <div spry:repeat="myDatabase"> ─3
        <div spry:choose="spry:choose"> ─4
```

```
      <span spry:when="'{color}' == '赤色'">{myDatabase::name}は
      {myDatabase::color}で{myDatabase::price}円</span>   ← 5
    </div>
   </div>
  </div>
 </body>
</html>
```

1 …Adobe Spryフレームワークを読み込む
2 …XMLデータを表示する領域を「spry:region」として指定する
3 …XMLデータを繰り返し処理する領域を「spry:repeat」として指定する
4 …特定のデータを抽出する領域であることを指定する
5 …「spry:when」を使って条件式を指定する

SOURCE CODE JavaScript main.js

```
myDatabase = new Spry.Data.XMLDataSet("data.xml", "/list/fruits"); ← 1
```

1 …読み込むXMLデータファイルと処理基準となる「XPath」を指定する

OnePoint spry:whenを使って条件を指定する

Adobe Sprｙフレームワークでは、XMLデータ内から特定の文字や値に一致するデータだけを表示することができます。特定の値に一致するデータだけを抽出するには、「<div spry:choose="spry:choose">」として抽出する領域を指定しておきます。その中で「spry:when」を使って条件式を指定します。

サンプルでは、「<span spry:when="'{color}' == '赤色'">」として<color>タグの内容が赤色の文字列の場合のみ表示するようにしています。タグの内容ではなく属性値を指定することもできます。属性値を指定する場合は、「タグ名/@属性値」として「/@」で区切って指定します。また、Adobe Spryには「if」などの条件判断も用意されているので、状況に応じて使い分けるとよいでしょう。

Column Adobe Spryにはエフェクト機能も装備されている

Adobe SpryはXMLデータやJSON形式などのデータ処理を得意としますが、データ処理ばかりではなく、次のエフェクト機能も用意されています。

- フェード処理を行う
- ブラインド処理を行う
- ハイライト処理を行う
- 拡大縮小処理を行う
- 左右に振動させる(シェイク)
- スライド処理を行う
- 潰し処理を行う

Adobe Spryのエフェクトサンプルを次のURLに用意してありますので、参考にしてみてください。

URL http://www.openspc2.org/reibun/Adobe_Spry/index.html

135 XMLデータをソートして表示する

ここでは、Adobe Spryフレームワークを使って、XMLデータを並べ替えて表示する方法について解説します。

JavaScript Sample - Windows Internet Explorer

果物の価格一覧

※ボタンをクリックすると、価格の安い順に並べ替えることができます。

ソート

リンゴは赤色で100円
ミカンは橙色で150円
メロンは緑色で400円
イチゴは赤色で210円
レモンは黄色で90円
サクランボは赤色で200円

[ソート]ボタンをクリックすると…

JavaScript Sample - Windows Internet Explorer

果物の価格一覧

※ボタンをクリックすると、価格の安い順に並べ替えることができます。

ソート

レモンは黄色で90円
リンゴは赤色で100円
ミカンは橙色で150円
サクランボは赤色で200円
イチゴは赤色で210円
メロンは緑色で400円

価格の安い順番にデータが表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script src="xpath.js" type="text/javascript"></script>
    <script src="SpryData.js" type="text/javascript"></script>   1
    <script type="text/javascript" src="main.js"></script>
```

```
  </head>
  <body>
    <h1>果物の価格一覧</h1>
    ※ボタンをクリックすると、価格の安い順に並べ替えることができます。
    <form action="./sort.cgi" method="get" name="mainForm">
      <input type="button" id="sortButton" value="ソート">
    </form>
    <div spry:region="myDatabase"> ―2
      <div spry:repeat="myDatabase"> ―3
        <span>{myDatabase::name}は{myDatabase::color}で
          {myDatabase::price}</span>円 ―4
      </div>
    </div>
  </body>
</html>
```

1 …Adobe Spryフレームワークを読み込む

2 …XMLデータを表示する領域を「spry:region」として指定する

3 …XMLデータを繰り返し処理する領域を「spry:repeat」として指定する

4 …表示するXMLデータのタグ名を指定する

SOURCE CODE JavaScript main.js

```
myDatabase = new Spry.Data.XMLDataSet("data.xml", "/list/fruits"); ―1
myDatabase.setColumnType("price", "number"); ―2
window.onload = function(){
  document.getElementById("sortButton").onclick = function(){
    myDatabase.sort("price") ―3
  }
}
```

1 …読み込むXMLデータファイルと処理基準となるXPathを指定する

2 …ソート対象とするデータの型を指定する(ここでは価格順にソートするので「number」(数値)を指定する)

3 …データをソートして表示する

OnePoint 「setColumnType()」で対象データの型を指定後に「sort()」を使ってソート処理を行う

Adobe Spryフレームワークでは、XMLデータを文字列・数値順にソートすることができます。ソート対象が数値の場合には、「setColumnType()」を使って対象となるタグの型を指定します。サンプルでは<price>タグを「number」(数値型)として設定しています。

ソート対象のデータ型を設定したら、「sort()」メソッドを使ってソート処理を実行します。このメソッドのパラメータにソート対象となるXMLタグ名を指定します。複数のタグ名でソートする場合には配列として指定します。2番目のパラメータはソート順を切り替えるかどうかを指定します。「sort()」メソッドを呼び出すたびに昇順と降順を切り替えるには、「"toggle"」の文字列を指定します。

136 HTMLテンプレートを利用する

ここでは、prototype.js ver 1.5を使ったHTMLテンプレートについて解説します。

JavaScript Sample - Windows Internet Explorer

天気情報

【南関東】
今日の天気はくもりです。最高気温は13℃です。

HTMLテンプレートを使ってデータが表示される

SOURCE CODE　HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="prototype.js"></script> ―1
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>天気情報</h1>
    <div id="message"></div>
  </body>
</html>
```

1 …prototype.jsライブラリを読み込む

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  var msec = (new Date()).getTime();
  new Ajax.Request("message.txt", { ─1
    method: "get",
    parameters: "cache="+msec,
    onSuccess:function(httpObj){
      var text = httpObj.responseText;
      var template = new Template(text);─2
      $("message").innerHTML = template.evaluate({ ┐
        weather : "くもり",                         ├3
        temp : "13℃"                               ┘
      });
    },
    onFailure:function(httpObj){
      $("tableData").innerHTML = "エラーで読み込めませんでした";
    }
  });
}
```

1 …テンプレートとなるテキストファイルを読み込む
2 …HTMLテンプレートとして生成する
3 …テンプレートの内容を指定されたデータに置換する

SOURCE CODE テキストファイル message.txt

```
【天気情報】<br>
今日の天気は#{weather}です。最高気温は#{temp}です。
```

OnePoint テンプレートを生成して「evaluate()」で置換する

prototype.jsライブラリのバージョン1.5以降には、HTMLテンプレート機能があります。この機能を利用すると、HTML文書内またはテキストファイル内の特定の部分だけをプログラムで置換することができます。置換対象となるのは「#{名前}」の文字列ですが、これは任意のパターンを指定することもできます。

HTMLテンプレート機能を利用するには、テンプレートとなるデータ・ファイルを「new Template()」として新たに生成します。生成したテンプレートオブジェクトの「evaluate()」でパラメータに置換するキーと値をプロパティリスト形式(連想配列(ハッシュ)形式)で指定します。

SECTION 137 別ファイルにデータを渡す

ここでは、別ページ(別のHTMLファイル)にデータを渡す方法について解説します。

送信する文字を入力してボタンをクリックすると…

別ページに入力された文字列が表示される

SOURCE CODE HTML index.htm(送信側)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>文字情報の送信</h1>
    <form action="./setcss.cgi" method="get" name="mainForm">
      送信文字:<input type="text" name="sendText" id="sendText"
      value="こんにちは"><br>
      <input type="button" id="sendButton" value="別ページに送信">
    </form>
  </body>
</html>
```

SOURCE CODE HTML next.html(受信側)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
```

```
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="receive.js"></script>
  </head>
  <body>
    <h1>文字情報の受信</h1>
    <div id="message"></div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("sendButton").onclick = function(){
    var text = document.getElementById("sendText").value; ―1
    location.href = "next.html?data="+encodeURIComponent(text); ―2
  }
}
```

1 …テキストフィールドに入力された内容を読み出す

2 …next.htmlファイルにクエリ文字列として連結しURLとする

SOURCE CODE JavaScript receive.js

```
window.onload = function(){
  var data = location.href.split("?")[1]; ―1
  var text = data.split("=")[1]; ―2
  document.getElementById("message").innerHTML =   ┐
  decodeURIComponent(text);                          ┘3
}
```

1 …受信した内容はURLに含まれているので「split("?")」として「?」以降のデータを取得する

2 …データはキーと値のペアなので「split("=")」としてキーと値を分離し値だけを読み出す

3 …データをデコードしページ上に表示する

OnePoint クエリ文字列として受信側のファイルに送信する

JavaScriptで他のページにデータを受け渡すには、受信先のファイルのURLにクエリ文字列として付加し、送信する方法があります。受信側のファイルでは、URLの「?」以降のクエリ文字列を取り出します。取り出した文字列からキーと値のペアを取り出し、必要なデータだけを抽出します。

クエリ文字列なので大量のデータを送信することはできませんが、いくつかの変数値などは送信することができます。また、日本語や記号などが含まれる場合には、必ずエンコードしてから送信します。

SECTION

138 グラフを表示する

ここでは、数値データをグラフ化して表示する方法について解説します。

SOURCE CODE　HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="prototype.js"></script>
    <script type="text/javascript" src="excanvas.js"></script>
    <script type="text/javascript" src="Plotr.js"></script>
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>産業別の前年度対比グラフ</h1>
    <div><canvas id="graph" height="300" width="600"></canvas></div>
  </body>
</html>
```

1 …必要なライブラリを読み込ませる

2 …グラフを描画するキャンバスタグを用意する

SOURCE CODE　JavaScript main.js

```
window.onload = function(){
  var dataset = {
    "year2005": [[0, 1], [1, 1], [2, 1.2], [3, 1.5]],
    "year2006": [[0, 0.5], [1, 2], [2, 2.15], [3, 4]]
  };
  var options = {
    padding: {left: 50, right: 0, top: 10, bottom: 10},
    backgroundColor: "#dbdbdb",
    colorScheme: "#550000",
    xTicks: [
      {v:0, label:"農業"},
      {v:1, label:"重工業"},
      {v:2, label:"工業"},
      {v:3, label:"情報"}
    ]
  };
  var bar = new Plotr.BarChart("graph",options);
  bar.addDataset(dataset);
  bar.render();
}
```

1 …グラフ化する数値データを用意する
2 …グラフの表示オプションを指定する
3 …グラフに表示するラベルを指定する
4 …棒グラフを生成する
5 …棒グラフにデータを追加する
6 …棒グラフを表示する

OnePoint Plotrライブラリを使ってグラフを描画する

JavaScriptでは、<canvas>タグをサポートしているブラウザ(Firefox1.5以降、Safari2、Opera9)の場合は、自由にグラフィックを描画することができます。また、Internet Explorerでもキャンバスの処理を動作させるためのライブラリが用意されています。これらを組み合わせてグラフを描画するのがPlotrライブラリです(Plotrライブラリを利用するにはprototype.jsライブラリのver 1.5以降が必要)。

グラフを表示するには、HTML文書内に<canvas>タグを指定しておきます。プログラム側では表示するためのグラフ値、グラフの色やラベルなどを指定します。棒グラフを表示する場合は、「Plotr.BarChart()」を使います。パラメータに<canvas>タグのIDと表示オプションを指定します。

表示するデータを設定し、「render()」メソッドで画面に描画を行います。

なお、PlotrライブラリIは、他にも折れ線グラフや円グラフなどをサポートしています。Plotrライブラリは、次のページからダウンロードすることができます。

URL http://www.solutoire.com/plotr

139 クッキーのデータを読み書きする

ここでは、クッキーデータを読み込んだり、書き込んだりする方法について解説します。

[書き込み]ボタンをクリックするとクッキーの書き込みを行う

書き込んだ後に[読み込み]ボタンをクリックすると…

書き込んだクッキーの内容が表示される

SOURCE CODE　HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; 
    charset=utf-8">
    <title>JavaScript Sample</title>
```

```
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>クッキーのデータの読み書き</h1>
    <form action="./cookie.cgi" method="get" name="mainForm">
      <input type="text" id="userData" name="userData"
      value="215"><br>
      <input type="button" id="writeButton" value="書き込み">
      <input type="button" id="readButton" value="読み込み">
    </form>
    <div id="result">結果:</div>
  </body>
</html>
```

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  document.getElementById("writeButton").onclick = function(){
    var data = document.getElementById("userData").value;
    cookie.write("userComment", data, 30);
  }
  document.getElementById("readButton").onclick = function(){
    var data = cookie.read("userComment");
    document.getElementById("result").innerHTML =
    "クッキーデータは「"+data+"」";
  }
}
var cookie = {
  write : function (theName__,theValue__,theDay__){ ―1
    if ((theName__ != null) && (theValue__ != null)){
      var expDay__ = "Wed, 01 Jan 2020 18:56:35 GMT";
      if (theDay__ != null){
        theDay__ = eval(theDay__);)
        var setDay = new Date();
        setDay.setTime(setDay.getTime()+(theDay__*1000*60*60*24)); ―2
        expDay__ = setDay.toGMTString();
      }
      document.cookie = ―3
        theName__ + "="+escape(theValue__)+";expires="+expDay__;
        return true;
    }
    return false;
  },
```

```
  read : function (theName__){ ―4
    theName__ += "=";
    theCookie__ = document.cookie+";"; ―5
    start__ = theCookie__.indexOf(theName__); ―6
    if (start__ != -1){
      end__ = theCookie__.indexOf(";",start__);
      return unescape(theCookie__.substring
      (start__+theName__.length,end__));       ―7
    }
    return false;
  }
}
```

1 …クッキーへの書き込みを行うメソッド

2 …クッキーの期限を指定する(クッキーデータの保存期限はGMT(グリニッジ標準時)にする必要がある)

3 …クッキーにデータを書き込む

4 …クッキーのデータを読み込むメソッド

5 …クッキーデータを読み込む

6 …指定された名前のデータを検索する

7 …データが存在する場合はデータ値を返し、そうでない場合は「false」を返す

OnePoint クッキーの読み書きは「document.cookie」にアクセスする

JavaScriptでは、データを保存するためのクッキーファイルへの書き込み、読み出しをサポートしています。クッキーファイルは、「document.cookie」で参照することができます。クッキーへの書き込みは、データ名とデータ値、そしてデータの保存期限を指定します。データの書き込み数には上限があるので、大量のデータ保存には向きません。また、恒久的に保存されるわけではありません。長期間にデータを保存する場合には、CGI経由でサーバーに保存する必要があります。

クッキーからの読み出しは、「document.cookie」に該当する名前のデータがあるかどうか「indexOf()」を使って調べます。名前があった場合には、データ値を取得します。

Column さまざまなJavaScriptライブラリ

AjaxによりJavaScriptが再評価、注目されたおかげで数多くのライブラリが作成されています。次のURLで数多くのライブラリにアクセスすることができます。

URL http://miniajax.com/
URL http://www.openspc2.org/JavaScript/Ajax/Library/index.html

また、クッキーに関するライブラリは次のURLにあります。

URL http://insin.woaf.net/code/javascript/cookiemanager.html
URL http://www.shiojiri.ne.jp/~openspc/js/

Break Time

オブジェクトへの参照を代入して実行速度を上げる

JavaScriptは、プログラムを実行するたびに命令を解釈するインタプリタ型の言語です。C言語やJava言語などのコンパイラ型言語では、コンピュータが解釈できるコード(マシン語)に変換して実行するため、速度が高速です。しかし、JavaScriptなどインタプリタ型言語は毎回命令を解釈するため、速度的には低速になります。

そこで、プログラムを工夫して実行速度を上げなければならない場面がでてきます。JavaScriptの場合、文書構造にも依存しますが、基本的にはオブジェクトへの参照を一度変数に入れておき、以後その変数を参照することで高速化することができます。特に「document.getElementById()」で参照する場合には、多くのブラウザで高速化できSafari2やOperaなどでは2倍ほど高速に処理が行われます。

```
window.onload = function(){
  s = new Date();
  for (var i=0; i<10000; i++){
    document.getElementById("result").innerHTML = 
    "Sample Text";
  }
  e = new Date();
  tm1 = (e-s) / 1000;

  s = new Date();
  obj = document.getElementById("result");
  for (var i=0; i<10000; i++){
    obj.innerHTML = "Sample Text";
  }
  e = new Date();
  tm2 = (e-s) / 1000;

  document.getElementById("time").innerHTML = tm1+"<br>"+tm2
}
```

CHAPTER - 12

GUI関連

140 ツリービューを表示する

ここでは、Yahoo UIライブラリを利用して、ツリービューを表示する方法について解説します。

「+」をクリックすると、ツリービューが展開される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" href="css/tree.css"
    type="text/css" media="all">
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="yahoo.js"></script>
    <script type="text/javascript" src="dom.js"></script>
    <script type="text/javascript" src="event.js"></script>
    <script type="text/javascript" src="treeview.js"></script>
    <script type="text/javascript" src="main.js"></script>
  </head>
```

```
  <body>
    <h1>リンク集</h1>
    <div id="treeData"></div>
  </body>
</html>
```

1 …必要なライブラリファイルを順番に読み込む

SOURCE CODE JavaScript main.js

```
var tree;
window.onload = function(){
  tree = new YAHOO.widget.TreeView("treeData");  ―1
  var root = tree.getRoot();  ―2
  var node1 = new YAHOO.widget.TextNode("検索エンジン", root, false);  ―3
  var cNode1 = new YAHOO.widget.TextNode("Google", node1, false);  ┐
  var cNode2 = new YAHOO.widget.TextNode("Yahoo!", node1, false);  ├4
  var cNode3 = new YAHOO.widget.TextNode("goo", node1, false);     ┘
  cNode1.href = "http://www.google.co.jp/";  ┐
  cNode2.href = "http://www.yahoo.co.jp/";   ├5
  cNode3.href = "http://www.goo.ne.jp/";     ┘
  tree.draw();  ―6
}
```

1 …ツリービューオブジェクトを生成する
2 …ツリービューの基準となるルートノードを取得する
3 …ルートノードにテキストを追加する
4 …「node1」のノードにテキストを追加する
5 …クリック時のリンク先を設定する
6 …ツリービューを描画する

OnePoint ツリービューのノードに表示テキストを追加する

ツリービューを表示するには、Yahoo UIライブラリを利用すると簡単です。まず、ツリービューに必要なライブラリファイルを読み込みます。ツリービュー表示に必要なファイルはyahoo.js、dom.js、event.js、treeview.jsです。実際に表示する部分には<div>タグのみ用意し、ID名を指定しておきます。

プログラムでは「YAHOO.widget.TreeView()」でツリービューオブジェクトを生成します。生成されたツリービューオブジェクトのルートノードを求めます。ツリービューでは表示されるテキストは親ノードを指定する必要があります。最初の項目はルートノードに連結する必要があります。

テキストノードの追加は、「YAHOO.widget.TextNode()」で行います。パラメータには追加するテキストと親ノード、階層化されている場合には、あらかじめ展開表示するかどうかを「true」または「false」で指定します。「true」を指定すると最初から展開表示され、「false」を指定すると展開表示されなくなります。

表示されるテキストがクリックされたときに他のページに移動させる場合には、追加したテキストノードの「href」プロパティに表示するページURLを指定します。設定が終了したら、「draw()」メソッドでツリービューを描画します。

141 アコーディオン表示を行う

ここでは、Adobe Spryフレームワークを利用して、アコーディオン表示する方法について解説します。

※Internet Explorer7では、パネルのタブの色が青く表示されます。なお、Internet Explorerの場合には、HTMLがTransitionalとそれ以外で挙動が変わります。

JavaScript Sample - Windows Internet Explorer
C:¥JavaScript-Sa
Live Search
JavaScript Sample
OSの種類
Windows
一番多く利用されているOSです。いくつかの種類があります。
Mac OS
UNIX
コンピュータ | 保護モード: 無効
100%

JavaScript Sample - Windows Internet Explorer
C:¥JavaScript-Sa
Live Search
JavaScript Sample
OSの種類
Windows
Mac OS
誰にでも手軽に扱えることを目指したOSです。
UNIX
コンピュータ | 保護モード: 無効
100%

アコーディオン表示され、パネルタブをクリックすると内容が開閉される

SOURCE CODE　HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" href="SpryAccordion.css"     ]1
    type="text/css" media="all">
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
2-  <script src="SpryAccordion.js" type="text/javascript"></script>
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>OSの種類</h1>
```

```
    <div id="Accordion1" class="Accordion"> ―3
      <div class="AccordionPanel"> ―4
        <div class="AccordionPanelTab">Windows</div> ―5
        <div class="AccordionPanelContent">
          一番多く利用されているOSです。いくつかの種類があります。 ―6
        </div>
      </div>
      <div class="AccordionPanel">
        <div class="AccordionPanelTab">Mac OS</div>
        <div class="AccordionPanelContent">
          誰にでも手軽に扱えることを目指したOSです。
        </div>
      </div>
      <div class="AccordionPanel">
        <div class="AccordionPanelTab">UNIX</div>
        <div class="AccordionPanelContent">
          古くからあるOSです。LinuxはUNIX互換OSです。
        </div>
      </div>
    </div>
  </body>
</html>
```

1 …アコーディオン表示に必要なスタイルシートファイルを読み込む

2 …アコーディオン処理を行うライブラリを読み込む

3 …アコーディオン全体を囲むタグを指定する(このID名がアコーディオン処理に必要になる)

4 …アコーディオンパネル領域を指定する

5 …アコーディオンのタブを指定する　6 …アコーディオンで表示する内容を指定する

SOURCE CODE　JavaScript main.js

```
window.onload = function() {
  var accObj1 = new Spry.Widget.Accordion("Accordion1"); ―1
}
```

1 …アコーディオンを描画/設定する

OnePoint Adobe Spryフレームワークを使ってアコーディオンを表示する

アコーディオン表示ができるライブラリはいくつかありますが、Adobe Spryフレームワークを使うと手軽に処理できます。アコーディオンを表示させるためには、全体を囲む<div>タグとパネル領域、パネルタブ、パネル内容に、それぞれ特定のスタイルシートのクラスを設定します。

HTML文書での指定が終われば、プログラムでアコーディオンオブジェクトを生成するだけで機能するようになります。これは「Spry.Widget.Accordion()」でパラメータにアコーディオン全体を囲むタグのIDを指定します。

SECTION

142 パネル表示を行う

ここでは、Adobe Spryフレームワークを利用して、パネル表示する方法について解説します。

ページが読み込まれると、パネルが表示される(パネルタイトルをクリックして展開/折りたたむことができる)

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" href="SpryCollapsiblePanel.css"
    type="text/css" media="all">                              1
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script src="SpryCollapsiblePanel.js"
    type="text/javascript"></script>                          2
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>用語解説</h1>
      <div id="CollapsiblePanel1" class="CollapsiblePanel">   3
```

```
      <div class="CollapsiblePanelTab">JavaScriptとは?</div>   4
        <div class="CollapsiblePanelContent">
        <p>JavaScriptはオブジェクト指向のスクリプト言語です。<br>
        10年目にして、やっと注目されるようにになりました。        5
        </p>
        </div>
      </div>
    </div>
    <div id="CollapsiblePanel2" class="CollapsiblePanel">
      <div class="CollapsiblePanelTab">Ajaxとは?</div>
        <div class="CollapsiblePanelContent">
        <p>AjaxはJavaScriptを利用して非同期通信を利用してデータをやりとりするものです。</p>
        </div>
      </div>
    </div>
  </body>
</html>
```

1 …パネル表示に必要なスタイルシートファイルを読み込む

2 …パネル処理を行うライブラリを読み込む

3 …パネル全体を囲むタグを指定する(このID名がパネル処理に必要になる)

4 …パネルタブを指定する

5 …パネルで表示する内容を指定する

SOURCE CODE JavaScript **main.js**

```
window.onload = function() {
  var accObj1 = new   1
Spry.Widget.CollapsiblePanel("CollapsiblePanel1");
  var accObj2 = new Spry.Widget.CollapsiblePanel
  ("CollapsiblePanel2", { contentIsOpen:false });   2
}
```

1 …パネルを生成し表示する

2 …パネルの最初の表示状態を閉じるようにする

OnePoint Adobe Spryフレームワークを使ってパネルを表示する

Adobe Spryフレームワークを使うと、手軽にパネル表示を処理できます。パネルを表示させるためには、全体を囲む<div>タグとパネルタブ、パネル内容に、それぞれ特定のスタイルシートのクラスを設定します。

HTML文書での指定が終わればプログラムでパネルオブジェクトを生成するだけで機能するようになります。これは「Spry.Widget.CollapsiblePanel()」でパラメータにパネル全体を囲むタグのIDを指定します。

SECTION 143 タブパネル表示を行う

ここでは、Adobe Spryフレームワークを利用して、タブパネルを表示する方法について解説します。

ページが読み込まれると、タブパネルが表示される（タブをクリックすると対応した内容に切り替わる）

SOURCE CODE　HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" href="SpryTabbedPanels.css"
    type="text/css" media="all">   1
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
```

```
    <script src="SpryTabbedPanels.js"
    type="text/javascript"></script>
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>プログラム言語の説明</h1>
      <div class="TabbedPanels" id="tabPanel1">
        <ul class="TabbedPanelsTabGroup">
          <li class="TabbedPanelsTab" tabindex="0">Perl</li>
          <li class="TabbedPanelsTab" tabindex="0">Ruby</li>
          <li class="TabbedPanelsTab" tabindex="0">Python</li>
        </ul>
        <div class="TabbedPanelsContentGroup">
          <div class="TabbedPanelsContent">CGIで多く使われています</div>
          <div class="TabbedPanelsContent">Perlより簡単手軽です</div>
          <div class="TabbedPanelsContent">
          Google社が積極的に利用しています</div>
        </div>
      </div>
  </body>
</html>
```

1 …タブパネル表示に必要なスタイルシートファイルを読み込む
2 …タブパネル処理を行うライブラリを読み込む
3 …タブパネル全体を囲むタグを指定する(このID名がタブパネル処理に必要になる)
4 …タブを指定する
5 …タブ内容を指定する

SOURCE CODE JavaScript main.js

```
window.onload = function() {
  var tbObj1 = new Spry.Widget.TabbedPanels("tabPanel1");
}
```

1 …タブパネルを生成して描画する

OnePoint Adobe Spryフレームワークを使ってタブパネルを表示する

Adobe Spryフレームワークを使うと、手軽にタブパネル処理を行うことができます。タブパネルを表示するには、全体を囲む<div>タグとパネルタブ、パネル内容にそれぞれ特定のスタイルシートのクラスを設定します。

HTML文書での指定が終われば、プログラムでパネルタブオブジェクトを生成するだけで機能するようになります。これは「Spry.Widget.TabbedPanels()」でパラメータにパネルタブ全体を囲むタグのIDを指定します。

SECTION 144 タブに表示する内容を別ファイルにして非同期で読み込ませる

ここでは、Yahoo UIライブラリを使ってタブに表示する内容を別ファイルにし、非同期で読み込ませる方法について解説します。

ページが読み込まれると、非同期で内容が読み込まれる（タブをクリックすると対応する内容に切り替わる）

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" href="tabs.css"
    type="text/css" media="all">
    <link rel="stylesheet" href="border_tabs.css"
    type="text/css" media="all">
    <link rel="stylesheet" type="text/css"
    href="main.css" media="all">
    <script type="text/javascript" src="yahoo.js"></script>
    <script type="text/javascript" src="dom.js"></script>
    <script type="text/javascript" src="event.js"></script>
    <script type="text/javascript" src="connection.js"></script>
    <script type="text/javascript" src="tabview.js"></script>
    <script type="text/javascript" src="main.js"></script>
  </head>
```

```
  <body>
    <h1>プログラム言語の説明</h1>
      <div id="informationTab" class="yui-navset"></div> ―3
  </body>
</html>
```

1 …必要なスタイルシートファイルを読み込む
2 …Yahoo UIライブラリを読み込む
3 …全体を囲むタグを指定する

SOURCE CODE　JavaScript main.js

```
window.onload = function(){
  var tabObj = new YAHOO.widget.TabView("informationTab"); ―1
  tabObj.addTab(new YAHOO.widget.Tab({ ―2
    label : " Perl ", ―3
    dataSrc : "./tab1.txt", ―4
    active:true ―5
  }));
  tabObj.addTab(new YAHOO.widget.Tab({
    label : " Ruby ",
    dataSrc : "./tab2.txt"
  }));
  tabObj.addTab(new YAHOO.widget.Tab({
    label : " Python ",
    dataSrc : "./tab3.txt"
  }));
  YAHOO.util.Event.onContentReady("doc", function() {  ┐
    tabView.appendTo("informationTab");                ┘6
  });
}
```

1 …タブビューオブジェクトを指定する
2 …タブを追加する
3 …タブに表示する文字を指定する
4 …読み込むテキストファイルのURLを指定する
5 …最初の表示するように指定する
6 …ページが読み込まれたらタブビューオブジェクトを構築する

OnePoint Yahoo UIライブラリのタブビューを使って非同期で読み込ませる

非同期でタブビューの内容を設定するには、Yahoo UIライブラリを使うと便利です。タブビューに必要なライブラリファイルであるyahoo.js、dom.js、event.js、connection.js、tabview.jsを順番に読み込ませます。

HTML文書内にはタブビューを表示するための<div>タグを用意し、クラス名とID名を指定しておきます。プログラムでは「YAHOO.widget.TabView()」でツリービューを生成し、「addTab()」で表示する内容を設定します。非同期で読み込ませる場合には「dataSrc」オプションに読み込むファイルURLを指定します。

設定が終わったら、「YAHOO.util.Event.onContentReady()」でページが読み込まれた際にツリービューを生成するようにします。

145 階層化メニューを表示する

ここでは、Adobe Spryフレームワークを使って、階層化メニューを表示する方法について解説します。

階層化メニューが表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; ⤶
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" href="SpryMenuBarHorizontal.css" ⤶   ─1
    type="text/css" media="all">
    <link rel="stylesheet" type="text/css" href="main.css" ⤶
    media="all">
    <script src="SpryMenuBar.js" type="text/javascript"></script> ─2
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>プログラム言語の基礎知識</h1>
    <ul id="MenuBar1" class="MenuBarHorizontal"> ─3
      <li><a class="MenuBarItemSubmenu" href="#">プログラム言語</a> ─4
      <ul>
        <li><a href="#">アセンブリ言語とは?</a></li>  ┐
        <li><a href="#">コンパイラとは?</a></li>      ├5
        <li><a href="#">インタプリタとは?</a></li>    ┘
```

```
        </ul>
      </li>
      <li><a class="MenuBarItemSubmenu" href="#">LL言語</a>
        <ul>
          <li><a href="#">LL言語とは?</a></li> ―6
          <li><a class="MenuBarItemSubmenu" href="#">他言語との比較</a>
            <ul>
            <li><a href="#">従来の言語</a></li>
            <li><a href="#">現在の言語</a></li>
            </ul>
          </li>
          <li><a href="#">Perl</a></li>
          <li><a href="#">Ruby</a></li>
          <li><a href="#">Python</a></li>
        </ul>
      </li>
    </ul>
  </body>
</html>
```

1 …必要なスタイルシートファイルを読み込む
2 …階層化メニューを表示するためのAdobe Spryライブラリを読み込む
3 …メニューバー全体を囲む領域を設定する
4 …メニューバー項目を指定する
5 …メニューを指定する
6 …階層化するには<ul>タグを入れ子にし、スタイルシートの「MenuBarItemSubmenu」クラスを指定する

SOURCE CODE JavaScript main.js

```
window.onload = function() {
  var hMenu1 = new Spry.Widget.MenuBar("MenuBar1"); ―1
}
```

1 …メニューバーを生成し表示する

OnePoint 階層化は<ul>タグを入れ子にして実現する

階層化メニューを表示するライブラリはいくつかありますが、基本的にはHTMLのタグ構造は同じです。リストタグである<ul>を入れ子にしてライブラリ固有のスタイルシートのクラスを指定します。

サンプルではAdobe Spryフレームワークを使って階層化メニューを実現しています。Adobe Spryではメニュー全体を囲むタグのスタイルシートのクラス名に「MenuBarHorizontal」、階層化メニューを表示するリストの<a>タグには「MenuBarItemSubmenu」クラスを指定するだけで実現することができます。

HTMLの設定が終わったら、プログラムでは「Spry.Widget.MenuBar()」としてメニューを構築します。パラメータにはメニュー全体を囲むID名を指定します。

SECTION

146 スライダーを表示する

ここでは、script.aculo.usライブラリを使って、スライダーを表示する方法について解説します。

JavaScript Sample - Windows Internet Explorer
http://192.168.1.
Live Search
JavaScript Sample
本書の評価
悪い←→良い
84%

スライダーをドラッグすると、割合が表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; 
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" type="text/css" href="main.css" 
    media="all">
    <script type="text/javascript" src="prototype.js"></script>
    <script type="text/javascript" src="scriptaculous.js"></script>
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>本書の評価</h1>
    <div>悪い←→良い
    <div id="sliderBG"> ―1
      <div id="pointer"></div> ―2
    </div>
    <div id="result"></div>
  </body>
</html>
```

1 …スライダー全体の領域を指定する

2 …スライダーのポインタを表示する領域を指定する

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  new Control.Slider("pointer", "sliderBG", { —1
    sliderValue : 0, —2
    onChange:function(value){ —3
      $("result").innerHTML = Math.floor(value * 100)+"%";
    },
    onSlide:function(value){ —4
      $("result").innerHTML = Math.floor(value * 100)+"%";
    }
  });
}
```

1 …スライダーオブジェクトを生成する。ポインタ領域とスライダー領域を指定する。
2 …スライダーの初期値を指定する　3 …スライダー値が変化した時に表示処理を行う
4 …スライダーがドラッグしている時にスライダー値の表示処理を行う

SOURCE CODE CSS main.css

```
body {
  background-color:#fff;
  color:#333;
  font-size:80%;
  line-height:1.8;
}
h1 {
  font-size:14pt;
  color:#777;
  padding:0;
  margin:0;
}
#pointer {
  width:5px;
  height:10px;
  background-image:url(point.gif);
  background-repeat:no-repeat;
}
#sliderBG {
  width:200px;
  height:4px;
  background-image:url(bg.gif);
  background-repeat:no-repeat;
}
#result {
  margin-top:20px;
}
```

OnePoint Control.Sliderでスライダーの生成と設定を行う

スライダーを表示するには、script.aculo.usライブラリを使うと便利です。script.aculo.usライブラリを利用するにはprototype.jsライブラリが必要なので、あらかじめ読み込ませておきます。

スライダーを表示するためにHTML文書内に<div>タグでスライダー全体の領域とポインタ領域を用意します。スクリプトで「Control.Slider()」のパラメータにスライダーのポインタと全体の領域を指定します。スライダーは、各種オプションが用意されています。スライダーの初期値は、「sliderValue」で設定することができます。スライダーが操作されると、「onSlide」イベントが発生するので対応する処理を指定することができます。スライダーの操作が終わったときに値が変化した場合には、「onChange」イベントが発生するので対応する処理を指定することができます。スライダー値は標準状態では0～1ですが、任意の値に変更したい場合にはオプションで「range:$R(0,100)」のように指定することができます。この場合、スライダー値の範囲は0～100までの数値になります。

Column Yahoo UIライブラリのスライダー

スライダー処理はscript.aculo.usライブラリだけでなく、Yahoo UIライブラリなどでもサポートされています。Yahoo UIライブラリの場合、縦横のスライダーと二次元のスライダーが用意されています。Yahoo UIライブラリを利用した横スライダーの場合、次のサンプルのようになります。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;charset=utf-8">
    <title>Sample</title>
    <link rel="stylesheet" href="main.css" type="text/css" ↩
    media="all">
    <link rel="stylesheet" href="logger.css" type="text/css" ↩
    media="all">
    <script type="text/javascript" src="yahoo.js"></script>
    <script type="text/javascript" src="dom.js"></script>
    <script type="text/javascript" src="event.js"></script>
    <script type="text/javascript" src="dragdrop.js"></script>
    <script type="text/javascript" src="slider.js"></script>
    <script type="text/javascript"><!--
      var slider;
        window.onload = function(){
          slider = YAHOO.widget.Slider.getHorizSlider ↩
          ("sliderBG", "sliderThumb", 0, 100, 5);
        }
      // --></script>
  </head>
  <body>
    <div id="sliderBG"><div id="sliderThumb"><img src="ptr.gif"></div>
    </div>
  </body>
</html>
```

147 カレンダーを表示する

ここでは、Yahoo UIライブラリを使ってカレンダーを表示する方法について解説します。

カレンダーが表示され、日付をクリックすると、その日付がページ上に表示される

SOURCE CODE HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html; 
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" href="css/calendar.css" 
    type="text/css" media="all">
    <link rel="stylesheet" type="text/css" href="main.css" 
    media="all">
```

```
    <script type="text/javascript" src="yahoo.js"></script>
    <script type="text/javascript" src="dom.js"></script>
    <script type="text/javascript" src="event.js"></script>
    <script type="text/javascript" src="calendar.js"></script>
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>カレンダー</h1>
    <div id="result"></div>
    <div id="myCalendar"></div>
  </body>
</html>
```

1 …Yahoo UIライブラリを読み込む
2 …カレンダーを表示する領域を指定する

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  var myCalendar = new YAHOO.widget.Calendar("cal", "myCalendar");
  myCalendar.render();
  myCalendar.selectEvent.subscribe(function(eventName, selectDate){
    var y = new String(selectDate).split(",")[0];
    var m = new String(selectDate).split(",")[1];
    var d = new String(selectDate).split(",")[2];
    document.getElementById("result").innerHTML = 
    "選択された日付:"+y+"年"+m+"月"+d+"日";
  }, myCalendar, true);
}
```

1 …カレンダーオブジェクトを生成してページ上に表示する
2 …日にちがクリックされたときに日付を表示する

OnePoint Yahoo UIライブラリのカレンダーを利用する

カレンダーを表示するには、Yahoo UI Libraryを使うと便利です。カレンダーを表示するために必要なライブラリファイル、yahoo.js、dom.js、event.js、calendar.jsを読み込みます。HTML文書ではカレンダーを表示する領域を<div>タグで指定します。

プログラムでは、「YAHOO.widget.Calendar()」としてカレンダーオブジェクトを生成します。カレンダーの表示は、「render()」で行います。

カレンダー上の日付が選択された場合に日付をページ上に表示するには、「selectEvent.subscribe()」として選択時のイベントを設定します。イベントハンドラには選択された日付が渡されるので、これを整形してページ上に表示します。

なお、Yahoo UIライブラリが用意しているカレンダーには、同時に2ヶ月分表示できるデュアルカレンダーもあります。

148 カラーピッカーを表示する

ここでは、カラーピッカーを表示する方法について解説します。

JavaScript Sample - Windows Internet E...
カラーピッカー
FF33FF

JavaScript Sample - Windows Internet E...
カラーピッカー
8AFF33

テキストフィールド上をクリックすると、
カラーピッカーが表示される

SOURCE CODE　HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;
    charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" href="colorPicker.css"
    type="text/css" media="all">
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
```

1

```
    <script type="text/javascript"
    src="lib/prototype.js"></script>
    <script type="text/javascript"
    src="lib/scriptaculous.js"></script>
    <script type="text/javascript"
    src="lib/yahoo.color.js"></script>
    <script type="text/javascript"
    src="lib/colorPicker.js"></script>
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>カラーピッカー</h1>
    <form>
      <input type="text" id="colorfield" value="FF33FF">
    </form>
  </body>
</html>
```

1 …カラーピッカーの表示に必要なスタイルシートファイルを読み込む

2 …カラーピッカーの表示に必要なライブラリファイルを読み込む

3 …テキストフィールドにID名を指定する

SOURCE CODE JavaScript main.js

```
window.onload = function(){
  new Control.ColorPicker("colorfield", { IMAGE_BASE : "img/" });
}
```

1 …カラーピッカーを生成する(この時点では表示されない)

OnePoint カラーピッカーを表示するライブラリを利用する

カラーピッカーを表示するには、次のライブラリを利用すると簡単です。

URL http://www.knallgrau.at/code/colorpicker

このライブラリはprototype.js、script.aculo.usを利用しているので、あらかじめ読み込ませておきます。HTML文書ではカラーピッカーを表示する時に必要となるテキストフィールドとID名を指定します。

プログラムでは、「Control.ColorPicker()」としてカラーピッカーを表示する際に読み出すテキストフィールドのID名を指定します。2番目のオプションパラメータには、ライブラリが使用するカラーピッカーの画像などが格納されているパスを指定します。

サンプルではカラーピッカーで色を指定すると、自動的にテキストフィールドの背景色/値も変化します。

149 ウィザード形式で画面を切り替える

ここでは、ウィザード形式で画面を切り替える方法について解説します。

[next]ボタンで次の画面に移動し、[previous]ボタンで前の画面に戻ることができます。

SOURCE CODE　HTML index.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type"
    content="text/html;charset=utf-8">
    <title>JavaScript Sample</title>
    <link rel="stylesheet" href="Wizard.css"
    type="text/css" media="all">
```

```
    <link rel="stylesheet" type="text/css" href="main.css"
    media="all">
    <script type="text/javascript" src="dojo.js"></script> —2
    <script type="text/javascript" src="main.js"></script>
  </head>
  <body>
    <h1>Macの起動方法</h1>
    <div id="wizardMac" dojoType="WizardContainer">—3
      <div dojoType="WizardPane">—4
        <h1>ステップ 1</h1>                                ┐
        <p>Macを起動するには電源ボタンを押します。</p>  ┘5
      </div>
      <div dojoType="WizardPane">
        <h1>ステップ 2</h1>
        <p>しばらくすると図のような画面が表示されます。アイコンをクリックします。</p>
      </div>
      <div dojoType="WizardPane">
        <h1>ステップ 3</h1>
        <p>パスワードを入力します。しばらくするとMacOS Xが起動します。</p>
      </div>
      <div dojoType="WizardPane" doneFunction="done">
        <h1>ステップ 4</h1>
        <p>ドック上のアイコンをクリックするとアプリケーションが起動します。</p>
      </div>
    </div>
  </body>
</html>
```

1 …ウィザード表示用のスタイルシートを読み込む

2 …Dojoライブラリを読み込む

3 …「dojoType」属性を使ってウィザードで使用するタグの範囲を指定する

4 …「dojoType」属性に「WizardPane」を指定し、1回に表示される領域(ペイン)を指定する

5 …ウィザード表示領域(ペイン)に表示する文字や画像などを指定する

SOURCE CODE JavaScript main.js

```
dojo.require("dojo.widget.Wizard");—1
dojo.hostenv.writeIncludes();

function done() {                              ┐
  alert("次はiPodを接続します");—3           │2
}                                              ┘
```

1 …Wizardウィジェットを読み込む

2 …ウィザード完了時の処理

3 …ウィザードが完了したときに呼び出される関数

OnePoint ウィザード形式の画面を構築するにはDojoのWizardウィジェットを利用する

ウィザード形式の画面を構築するには、DojoライブラリのWizardウィジェットを利用すると簡単にできます。ウィザード画面全体を囲む<div>タグを用意し、「dojoType」属性にWizardContainerを指定します。次に1画面ごとの表示領域を<div>タグで囲み、「dojoType」属性に「WizardPane」を指定します。これは、切り替える画面の数だけ用意します。

最後に処理が完了したときに呼び出す関数がある場合には、最後の画面を表示する<div>タグに「doneFunction」属性を指定します。「doneFunction」属性値に処理完了時に呼び出す関数名を指定します。

HTMLの設定が終わったらプログラムで「dojo.require("dojo.widget.Wizard")」としてスクリプトファイルを読み込ませます。これだけで自動的にウィザード処理が行われます。

Column Dojo ToolKit

数あるAjax/JavaScriptライブラリの中で、もっとも巨大なのがDojoです。多くのライブラリは、特定の処理に特化しています。たとえば、prototype.jsであればJavaScriptのコア機能の拡張、script.aculo.usであればエフェクト、Yahoo UI Libraryであればユーザーインターフェース、Adobe SpryであればXMLデータ処理といった具合です。特化された機能を持つライブラリを組み合わせて目的の処理を行いますが、その際に問題になるのがライブラリの干渉です。つまりライブラリの組み合わせによっては誤動作を引き起こしてしまうことがあります。

このような障害を防ぐには、prototype.jsに依存するライブラリだけで構成するといった工夫が必要になります。このような組み合わせによる障害は、Dojoのようにすべてをサポートするライブラリがあれば解消されます。

しかし、すべての機能を持つようにするため、どうしてもライブラリ自体が巨大化してしまいます。ライブラリが巨大化/肥大化すると読み込みまでに時間がかかったり、環境によってはメモリ不足を引き起こすこともあります。また、実行速度も低下してしまうこともあります。

Dojoにはサンプルで掲載したウィザード機能だけでなく、実にさまざまな機能を提供しています。詳しくは次のURLを参照してください。

URL http://dojotoolkit.org/

●The Dojo Toolkit

Break Time

デバッガを使って不具合を解決する

Ajaxの登場以前はJavaScriptのデバッガは、あまり良いツールがありませんでした。このため、デバッグコンソールやステータスバー、「alert()」を使ってデバッグするという状況でした。しかし、現在では、「Firebug」など高性能なデバッガがあるので、これらを使用するのがよいでしょう。

特にFirebugは非同期通信の状況や、どのような内容で読み込まれたか、DOMの状態やスタイルシートがどのように上書き設定されたかなど非常に詳細な情報を表示してくれます。また、スタイルシートのプロパティを追加したり、値をリアルタイムに変更することができます。

Firebugは、次のページでダウンロードすることができます。

● **Firebug**

URL https://addons.mozilla.org/firefox/1843/

◉Firebugの入手先

INDEX

記号

A・B・C・D

E・F・G・H

I・J・L

M・N・P・R

S・T・U

V・W・X・Y

あ・か・さ行

た・な・は行

ま・や・ら行

■著者紹介

古籏　一浩（ふるはた　かずひろ）

この本で27冊目です。30冊まで、あと少し。今回は初めて完全にHTML文書とJavaScriptコードを分離して作成しましたが、どうもこの方が早く作業できるようです。また、不具合も少ない感じです。

URL　http://www.openspc2.org/

E-mail　openspc@po.shiojiri.ne.jp

編集担当：吉成明久　　カバーデザイン：秋田勘助（オフィス・エドモント）

JavaScriptテクニックブック

2007年5月7日　初版発行

著　者　古籏一浩

発行者　池田武人

発行所　株式会社　シーアンドアール研究所

本　　社　新潟県新潟市北区西名目所 4083-6（〒950-3122）

営業本部　東京都千代田区飯田橋 3-2-12-201（〒102-0072）

電話　03-3288-8481　　FAX　03-3239-7822

印刷所　株式会社　ルナテック

ISBN978-4-903111-52-0 C2055

ISBN978-4-903111-52-0

C2055 ¥2500E

9784903111520

1922055025007

C&R研究所

定価：本体2,500円＋税

すぐに使える

JavaScriptテクニックブック

```
window.onload = function(){

document.getElementById("cre
ateButton").onclick =
function(){
  var dateObj = new Date();
  var h = dateObj.getHours();
  var m =
dateObj.getMinutes();
  var s =
dateObj.getSeconds();
  var text =
document.createTextNode(h+":
"+m+":"+s+"");

document.getElementById("res
ult").appendChild(text);
```

```
window.onload = function(){
 window.document.onkeydown =
function(evt){
  if (evt){
   var kc = evt.keyCode;
  }else{
   var kc = event.keyCode;
  }
  var chr = String.fromCharCode(kc);

document.getElementById("result").in
```

```
window.onload = function(){
 setInterval("album.swapImage()"
2000);
}
var album = {
 imageURL : [
      "images/01.jpg",
```

```
        swapImage : function(){
  this.count++;
  if (this.count >= this.imageURL.length)
this.count = 0;
  document.getElementById("photo").src =
this.imageURL[this.count];
 },
 count : 0
};
```

JavaScript technic book

CHAPTER-01 JavaScriptの基礎知識
CHAPTER-02 基本テクニック
CHAPTER-03 フォーム
CHAPTER-04 画像
CHAPTER-05 ダイアログ/ウィンドウ
CHAPTER-06 DOM
CHAPTER-07 スタイルシート
CHAPTER-08 イベント
CHAPTER-09 文字列
CHAPTER-10 日付/時刻
CHAPTER-11 その他
CHAPTER-12 GUI関連

古籏一浩／著

C&R研究所